TESINA D'ESPECIALITAT

Títol

Implementation and Testing of Two Bee-Based Algorithms in Finite Element Model Updating

Autor/a

Roser Marrè Badalló

Tutor/a

Rolando Chacón Flores

Departament

Enginyeria de la Construcció

Intensificació

Anàlisi i Projectes d'Estructures

Data

Octubre 2013

Contents

List of Figures

List of Tables

Preface

This Master Thesis was carried out at the *Department of Civil and Architectural Engineering*, the division of *Structural Engineering and Bridges*, at KTH *Royal Institute of Technology* in Stockholm, during 5 months.

My gratitude to Professor Raid Karoumi, who gave me the idea and opportunity to develop this Master Thesis, and to Ignacio González, who supervised it, for helping and guiding me through all the process.

Thanks to laboratory technician Claes Kullberg, who managed the instrumentation and helped me in the experimental part.

Many thanks also to Professor Rolando Chacón, who helped, encouraged and advised me from the distance.

Thanks also to my friends from Kista, who made me feel like home. Thanks to my family for their unconditional love, support and encouragement during all my life and especially during the realization of this Master Thesis, and to Arnau for his patience and to believe in me.

In general, thanks to all those who have influenced directly or indirectly on the realization of this Master Thesis.

Stockholm, June 2013

Roser Marrè Badalló

Abstract

Finite Element Model Updating has recently arisen as an issue of vast importance on the design, construction and maintenance of structures in civil engineering. Many algorithms have been proposed, developed and enhanced in order to accomplish the demands of the updating process, mainly to achieve computationally efficient programs and greater results.

The present Master Thesis proposes two new algorithms to be used in Finite Element Model Updating: the Bees Algorithms (BA) and the Artificial Bee Colony algorithm (ABC). Both were first proposed in 2005, are based on the foraging behaviour of bees and have been proved to be efficient algorithms in other fields.

The objective of this Master Thesis is, thus, to implement and to test these two new algorithms in Finite Element Model Updating for a cantilever beam. The Finite Element Model and the algorithms are programmed, followed by the extraction of the experimental frequencies and the updating process.

Results, comparison of these two methods and conclusions are given at the end of this report, as well as suggestions for further work.

Keywords: Model Updating, Finite Element Method, Bees Algorithm (BA), Artificial Bee Colony (ABC).

Sammanfattning

Finite Element Model Updating (Finitelement Modelluppdatering) har nyligen fått stor betydelse för utformning, konstruktion och underhåll av strukturer för infrastruktur. Många algoritmer har föreslagits, utvecklats och förbättrats i syfte att uppnå främst beräkningsmässigt effektiva program och bättre resultat.

Detta examensarbete föreslår två nya algoritmer som skulle kunna användas i Finite Element Model Updating: Bees Algorithm (Bin Algoritm) och Artificial Bee Colony algorithm (Artificiella Bikoloni algoritm). Båda först föreslogs år 2005, baserat på samligsbeteende av bin och har visat sig att vara effektiva algoritmer även inom andra områden.

Målet med detta examensarbete är alltså att implementera och testa dessa två nya algoritmer i Finite Element Model Updating för en konsolbalk. Finita Element Modellen och algoritmerna programmerats, följt av mätning av de experimentella frekvenserna och till sist uppdateras modellen med hjälp av de föreslagna algoritmer.

Resultaterna av dessa två algoritmer visas och jämförs.

Nyckelord: Modelluptatering, Finitelement Metoden, Bin Algoritm, Artificiella Bicolony algoritm.

Resum

El Finite Element Model Updating (Ajustament o Actualització de Models en Elements Finits) ha sorgit recentment com un tema de gran importància en el disseny, construcció i manteniment d'estructures en enginyeria civil. Molts algoritmes han estat proposats, desenvolupats i millorats per tal de complir les exigències del procés d'actualització, principalment per aconseguir programes computacionalment eficients i millors resultats.

La present Tesina de Final de Carrera proposa dos nous algoritmes per ser utilitzats en Finite Element Model Updating: el Bees Algorithm (Algoritme de les Abelles) i l'Artificial Bee Colony algorithm (algoritme de la Colònia Artificial d'Abelles). Ambdós foren proposats per primera vegada el 2005, es basen en el comportament de les abelles quan busquen aliment i han demostrat ser uns algoritmes eficients en altres camps.

L'objectiu d'aquesta Tesina és, per tant, implementar i provar aquests dos nous algoritmes en Finite Element Model Updating en una biga en voladiu. Es programen el Model d'Elements Finits i els algoritmes, seguits per l'extracció de les freqüències experimentals i del procés d'actualització.

Es donen resultats, comparació d'aquests dos algoritmes i conclusions al final d'aquest article, així com suggeriments per noves línies d'investigació.

Paraules clau: Ajustament/Actualització de Models, Mètode dels Element Finits, Algoritme de les Abelles, algoritme de la Colònia Artificial d'Abelles.

Chapter 1

Introduction to Finite Element Model Updating

1.1 Introduction

1.1.1 Structural dynamics

Structural dynamics is the analysis of the behaviour of structures when subjected to a changing -position and magnitude- loading within a short time. This study embraces many fields, for example mechanical, aerospace or civil engineering. The present Master Thesis will be focussed on the last one, specially on structures such as bridges, dams or buildings which should be able to withstand from small vibrations of human steps, cars and winds to severe dynamic loading of earthquakes or hurricanes.

Unexpected behaviours of structures have occurred in the past, for instance:

- *The Tacoma Bridge:* after four months of being opened, in 1940, its main span began to have increasing torsional oscillations due to a 67 km/h wind and it finally collapsed (Green & Unruh, 2006).
- *The Hansin Expressway:* some parts of it collapsed during the Kobe earthquake, in 1995. It was believed to be earthquake proof, but it fell onto its side (Gazetas et al., 2006).
- *The Millennium Bridge:* when it was first inaugurated, in 2001, the horizontal frequency of human steps synchronized with a natural frequency of the bridge. This lead to resonance and it had to be closed in order to make some modifications (Strogatz et al., 2005).

Therefore, the prediction of dynamic behaviour of structures subjected to some loads so that undesired situations are avoided is of great importance. The examination of the dynamics of a structure can be accomplished by the use of numerical models.

1.1.2 Numerical models. Finite Element Modelling

A numerical model is a mathematical representation -by means of equations- of the behaviour of an "object" -in this case, a structure- which uses a stepping procedure in order to obtain its response -solving the equations-.

Some numerical models could be solved manually for simple structures. But, unfortunately, an analytical solution of the equation of motion for the system is usually not possible if the excitation varies arbitrarily with time or if the system is non-linear. Such problems should, then, be solved using numerical time-stepping methods for integration of differential equations, using for example Finite Element Modelling.

The Finite Element Method (FEM) is a discretization technique which subdivides the mathematical model into simpler components, called elements, which do not overlap in space. The response of each element is expressed in terms of a finite number of degrees of freedom (DOF) characterized as the value of an unknown function, or functions, at a set of nodal points. Therefore, the response of the mathematical model is considered to be approximated by an assembly of the discrete responses of all elements.

The tool often used in order to program the FE models is MATLAB (Kwon & Bang, 2000), which is a numerical computing environment and programming language developed by MathWorks, but it is not practical for large problems. For complex structures, more sophisticated general purpose programs such as ANSYS (Moaveni, 2003), ABAQUS (Börgesson, 1996) or LUSAS (LUSAS, 2010) should be used.

Advantages and disadvantages of FEM

There are three main advantages of creating numerical models:

1. The behaviour of a non-built structure subjected to loading can be predicted, which will lead to a safe and optimized built structure.

2. The characteristics of the model can be changed without costs. Numerical experiments can be done by trial and error until the desired behaviour is achieved.

3. Changes and damages in already built structures can be identified. This field of study inside model updating is called *damage detection* and is beyond the scope of the present Master Thesis.

Nevertheless, there is one main disadvantage of these models: they usually cannot represent the real behaviour of the structure accurately enough. This is due to different kind of errors (Mottershead & Friswell, 1993):

- *Model structure errors*, occurring when there is uncertainty on the governing physical equations. This might be the difficulty of modelling damping, joints, welds and edges.

- *Model parameter errors*, which result in the difficulty in identifying the correct material properties, as well as the application of inaccurate assumptions to simplify the model and inappropriate boundary conditions.
- *Model order errors*, appearing in the discretization of complex systems, for instance from the difficulty in modelling non-linearity.

In order to be able to detect the possible errors emerged from the numerical modelling, measuring the real behaviour of the structure and its dynamic properties -such as damping and natural frequencies- has gained vital importance. It is the introduction of the experimental data to the numerical model what makes the last one more realistic.

1.1.3 Experimental data

As mentioned in the previous Section, measurements are of great importance when it comes to modelling a structure. One might even think that models could be avoided if measurements are possible and/or easy to measure, but this is usually not the case: they are usually imprecise and incomplete. Experimental data should, then, be a complement of the numerical model, as the last one saves money, predicts and optimizes the features of structures.

1.1.4 Model Updating

In the previous Sections, the need of both numerical models and experimental data is explained. Yet, there is as well a need of a tool which is able to interconnect the numerical model and the experimental data: model updating.

Model updating is defined as "the process of correcting the numerical values of individual parameters in a mathematical model using data obtained from an associated experimental model such that the updated model more correctly describes the dynamic properties of the subject structure" (Ewins, 2000a). In other words, it is the procedure of adjusting the finite element model in order to fit the experimental data as closely as possible -assuming that the measured data is accurate and available-, making use of one algorithm or a combination of them. Thus, model updating tries to change the uncertain parameters so that the errors which they produce can be minimized.

To summarize, the essentials needed for model updating are: a finite element model with some uncertain parameters which need to be updated, data gathered in experiments and some algorithms which will update the numerical model so that its response matches the experimental data.

1.2 Motivation

Since the 1900s, Finite Element Model Updating has arisen as an issue of vast importance on the design, construction and maintenance of structures in civil engi-

neering. In all these three situations, undesired circumstances can be avoided and money can be saved using Model Updating: on the first one, the numerical models are compared to similar already built structures in order to detect possible problems in their dynamics; on the second one, a regular control of the structure is carried out while being built; and on the third one, damage in structures -or a lack of it- is detected and repaired, allowing the structures to be in service for more years.

Many different algorithms exist and are being developed in order to accomplish the purpose of model updating. Moreover, there is nowadays a need of obtaining computationally efficient programs which are able to guarantee improved results, as the importance of structures with a large amount of degrees of freedom is growing.

Consequently, it is important to continue exploiting the current algorithms and enhancing their features, as well as studying new algorithms to implement, and, with that, obtain refined results, even with complex structures. Model updating is a science which is still being developed and which is essential to keep studying.

1.3 Objectives

General objective:

The main objective is to implement and compare two new algorithms in the process of Model Updating: Bees Algorithm (BA) and Artificial Bee Colony algorithm (ABC). Both algorithms are based on the foraging behaviour of honey bees, were first proposed in 2005 and are currently used for the optimization of multi-variable functions. The first one was recently and only tested in Model Updating by Moradi et al. (2010) and the second one has never been implemented in Model Updating. Hence, two "new" possibilities are analysed in Model Updating.

Specific objectives:

In the present Master Thesis, an analysis of the processes involved in Model Updating will be performed. The study will try to answer the following questions:

- Which have been and are the most commonly used algorithms in model updating?
- How feasible are the two implemented algorithms using both simulated and experimental data?
- Which algorithm performs better in the given conditions?

These questions will lead to a process of updating which will consist of:

1. Programming a numerical model in MATLAB of a simple steel structure (cantilever beam).
2. Programming and validating the two algorithms in MATLAB.

3. Testing the two algorithms with simulated data.

4. Arranging the real model.

5. Determining the natural frequencies of the real model by means of experiments.

6. Updating the numerical model with the two algorithms separately.

1.4 Methodology

In Chapter 2 (Theory related to Model Updating), necessary bases on the process previous to model updating are explained. These include: the equation governing the dynamics of a structure in order to numerically model it, considerations on the numerical model which is to be updated, limitations of experimental tests and how data is processed, and methods to compare numerical data -which is going to be updated- with experimental tests.

Subsequently, a state-of-the art of the most relevant updating algorithms used in the past and currently used are summarised in Chapter 3 (Model Updating methods). This chapter also includes their classification, advantages and disadvantages, and applications.

A description of both algorithms to be implemented -Bees Algorithm and Artificial Bee Colony algorithm- is performed afterwards, in Chapter 4 (Recent bee-based algorithms), which includes a previous background of the behaviour of bees in order to be able to understand these two algorithms.

Chapter 5 (Technical description of the programs) comprises a description of all the programming part and how the programs are computed. Moreover, it includes the validation of them, presenting motives of the chosen control parameters, which are tested with simulated data.

In Chapter 6 (Experimental part), all information related to gathering the experimental data is presented. It includes a description of the implementation of the experimental model, the programs and tools required for the measurements and how the data is transformed and finally analysed in order to extract the necessary information.

The following, Chapter 7 (Results and analysis), comprehends the updating part of the numerical model using the data gathered in the previous chapter. The results obtained using both algorithms are presented, analysed and compared.

Finally, Chapter 8 (Conclusions and future work) gathers the conclusions extracted from the Present Master Thesis, as well as recommendations and future work that should be done in order to complement this study.

1.5 Scope and limitations

Updating is a process with numerical difficulties, which are derived from the inaccuracy in the model, and imprecision and insufficient information in the measurements.

In this Master Thesis, some assumptions are made, which lead to limitations:

- The study is focussed on the performance of BA and ABC on a cantilever beam.
- The design and verification of the resistance of the beam are beyond the scope. The usual characteristics of a steel beam are considered.
- A 2D Euler-Bernoulli beam is assumed in the numerical model. Therefore, a beam which is only subjected to vertical load and without shear deformation is considered.
- Although the experiments show a mitigation of the vibration in time, no damping is considered in the numerical model in order to simplify the problem, as most real civil engineering structures are lightly damped.
- Even though the experiments are not carried out in the best conditions, it is assumed that the experimental measurements are correct. Thus, the aim of Model Updating will be to make improvements in the Finite Element Model so that the updated response matches the measured data in vibration tests as accurately as possible.

Chapter 2

Theory related to Model Updating

This Chapter contains the necessary bases on the previous process to model updating, which need to be mentioned in order to be able to carry out all the processes involved in Finite Element Model Updating. These processes are:

1. Programming the model.

2. Gathering the experimental data.

3. Comparing the data obtained from the numerical model with the experimental data.

Therefore, regarding the first step, the first section of this chapter includes what the numerical model is based on and what should be kept in mind when programming a numerical model. Regarding the second one, disadvantages of gathering the experimental data and what domains it is represented in. Finally, regarding the last step, it explains how experimental and analytical data should be compared before updating the numerical model.

This Chapter should not be considered as deeply explicative, but simply as an introductory chapter. All the information included in the present Chapter will be used in subsequent chapters which will make this information easier to understand.

2.1 Numerical Modelling

2.1.1 Equation of motion

The equations of motion are mathematical expressions defining the dynamic displacements of a structure during a period of time and are based on Newton's second law of motion. In other words, the equation of motion is the numerical model of the structure and its solution represents the theoretical response of it. All the equations given in this section can be obtained from Chopra (2001).

Assuming the behaviour of the structure as linear, the general equation of motion in finite elements is the following:

$$[M] \cdot \{\ddot{u}\} + [C] \cdot \{\dot{u}\} + [K] \cdot \{u\} = \{F\} \tag{2.1}$$

Where:

- $[M]$, $[C]$ and $[K]$ are the mass, damping and stiffness matrices, respectively.
- $\{u\}$, $\{\dot{u}\}$ and $\{\ddot{u}\}$ are the displacement, velocity and acceleration vectors, respectively.
- {F} is the applied force vector.

The dimension of these matrices and vectors is nxn and nx1, respectively; being n the degrees of freedom of the modelled structure. These matrices will be defined later in Chapter 5.

Many structures in civil engineering are lightly damped and considering them undamped in the finite element model may not affect considerably in their response. Thus, neglecting the damping of the structure, $[C] = [0]$, Equation 2.1 becomes:

$$[M] \cdot \{\ddot{u}\} + [K] \cdot \{u\} = \{F\} \tag{2.2}$$

In the study of structural dynamics, the free vibration is highly important, which is the movement of the structure when is not subjected to any kind of external load. In this case, making the external load equal to zero, $\{F\} = \{0\}$, in Equation 2.2:

$$[M] \cdot \{\ddot{u}\} + [K] \cdot \{u\} = \{0\} \tag{2.3}$$

It is known from studying the real behaviour of structures in experimental tests that the free vibration of a structure is harmonic. Therefore, an harmonic approach of the displacement for each degree of freedom can be assumed:

$$u = U \cdot e^{st}$$

Where s are the unknowns, U is a constant related with the amplitude and t is the time. In the specific case where $DOF = 1$ and the system is undamped, the solution results in:

$$u = C \cdot \sin(\omega_n \cdot t - \alpha)$$

Here C is a constant related also with the amplitude, ω_n is the natural circular frequency and α is the initial phase angle. Deriving two times this expression:

$$\dot{u} = \omega_n \cdot C \cdot \cos(\omega_n \cdot t - \alpha) \rightarrow \ddot{u} = -\omega_n^2 \cdot C \cdot \sin(\omega_n \cdot t - \alpha) = -\omega_n^2 \cdot u$$

For more than one DOF, the same approach can be adopted and, therefore, Equation 2.3 can be expressed as:

$$\Big([K] - \omega_n^2[M]\Big) \cdot \{u\} = \{0\} \tag{2.4}$$

This type of problem is very common and it is called an eigenvalue problem. It has the following form:

$$([A] - \lambda_n[I]) \cdot \{v\} = \{0\}$$

$[A]$ is a known matrix, $[I]$ is the identity matrix, $\{v\}$ is a vector and each λ_n is an unknown value. The aim is to solve the equation in order to obtain λ_n -called eigenvalues- and, with that, obtain $\{v\}$ -called eigenvectors-. For systems with a low number of DOF it can be solved manually, but for higher systems MATLAB or other computing programs can be used.

When solving Equation 2.4, the modal properties of the structure are obtained. These are the natural frequencies -eigenvalues- and the displacements -eigenvectors- or also called as mode shapes of the structure. While the number of degrees of freedom increases, so does the number of frequencies obtained, which must be the same. As changes in the mass and stiffness matrices cause modal properties to be also changed, the modal properties can be extracted from the identification of the correct mass and stiffness matrices.

Chapter 5 will give further details on how this numerical model is programmed.

2.1.2 FE modelling in model updating

A numerical model which will be updated needs to have into account some factors that would not be necessary to consider if the model was not going to be updated. The most important factor is to choose the right updating parameters, which are the parameters inside the mass and stiffness matrices of the equation of motion, or the boundary conditions. It should be analysed which of these parameters from the numerical model are more reliable and which ones are less. For example, the middle of a beam -away from the boundaries- could be modelled with confidence; while joints and constraints might be less accurate and should be considered for updating.

It is important that the numerical predictions -natural frequencies and mode shapes- are sensitive to small changes in these parameters. Otherwise, the updating procedure will result in a change of other parameters more sensitive but possibly in less need of updating. As a result, the updated model would lack of physical meaning (Mottershead & Friswell, 1993).

2.2 Vibration Testing

Depending on the accuracy and quantity of information extracted from the test measurements of a structure, the numerical model can be more or less improved by updating its parameters. Usually, the measurements are:

Imprecise. This is due to:

- Noise: unwanted random data without meaning that causes distortion to measurements. It comes both from the instruments or from signal processing, and can be minimized using transducers, filters and windows.
- Systematic errors: a biased tendency in measurements in relation to the actual value. They can appear when the boundary conditions are not well-replicated, when the mass of the accelerometer changes the mode shapes or the frequencies of the structure, or when the measurement instruments are not well-calibrated.

Incomplete. The problem resides in that the number of measured degrees of freedom does not correspond to the one in the Finite Element Model. Therefore, two approaches can be implemented in order to minimize the problem:

- The experimental data may be expanded to the set of Finite Element Model degrees of freedom. This will introduce errors in the data required by the updating algorithm, as the Finite Element Model may have errors.
- The Finite Element Model may be reduced to the measured degrees of freedom. This might imply that changes in the updated model would be lying in a large number of degrees of freedom.

The procedure of measuring consists of three stages:

- Excitation of the structure.
- Measurement of the acceleration response with an accelerometer.
- Data acquisition and processing: the data is amplified, filtered, converted from analogue to digital format and stored in a computer.

Once the measured data is converted to a digital signal it might be processed by computer hardware. At this time, the signal is in the time domain. Nevertheless, time, frequency and modal domains are the three types of domains, the possible ways of analysing the measured data. No information is lost in changing from one domain to another, it is just a change of perspective.

Time-domain data: It is the traditional way, a record of what happened to a parameter of the system during a period of time. In the x-axis there is time and in the y-axis the amplitude of the signal.

Frequency-domain data: Any wave form that exists in the real world can be generated by making a unique combination of sine waves, which will have their own frequency. Therefore, the frequency-domain data is the one which represents these frequencies in the x-axis and amplitudes in the y-axis. This representation is called the spectrum of the signal and each sine wave line is called a component of the signal.

Frequency-domain data are very useful, as the small components of the signal -distortion or noise- can be easily distinguished, and is the one used in the present Master Thesis.

Modal-domain data: The modal-domain data are expressed as natural frequencies, damping ratios and mode shapes. The most commonly used technique for extracting the modal properties is the modal analysis (Ewins, 2000b). The aim of modal-domain data is to detect changes in one of these three parameters in order to identify changes in the structure.

2.3 Comparing numerical data with test results

The most important step in model updating is to make a comparison between the properties of the numerical model and the ones from the test. If the degree of similarity between these two sets of data is not successful, the process of updating the model is very unlikely to succeed. This means that it is vital to obtain a satisfactory similarity in order to obtain a good updating process.

There are basically two main ways of comparing data: comparing modal properties or comparing frequency response functions (FRF).

A frequency response function is a mathematical representation of the relationship between the input and the output of a system and, thus, it describes the response of a structure to an excitation as a function of frequency. Nevertheless, the comparison of FRF is beyond the scope of the present Master Thesis and will not be explained.

Different methods of comparing modal properties of the numerical model with the data gathered will be explained (Ewins, 2000c) in the following section. Naturally, not all the existing methods will be covered, some other methods can be found in Allemang (2003).

2.3.1 Comparison of modal properties

The advantage of modal properties is that they can be predicted individually and, therefore, the ones specifically chosen can be compared with the test data easier. Nevertheless, the comparison of modal properties requires these modal properties to be extracted from the test, but this disadvantage does not prevent it from being the most common methodology.

Comparison of natural frequencies

This is the simplest method, comparing the measured frequencies with the theoretical ones in order to identify the degree of similarity between the pairs of frequencies and the possible cause of their differences. It can be done in a simple tabulation of the two sets of results, but it is more useful to plot a graph with the experimental and theoretical frequencies for each of the available modes. In this graph, the x-axis

represents the measured frequencies and the y-axis the theoretical ones, as shown in the example of Figure 2.1.

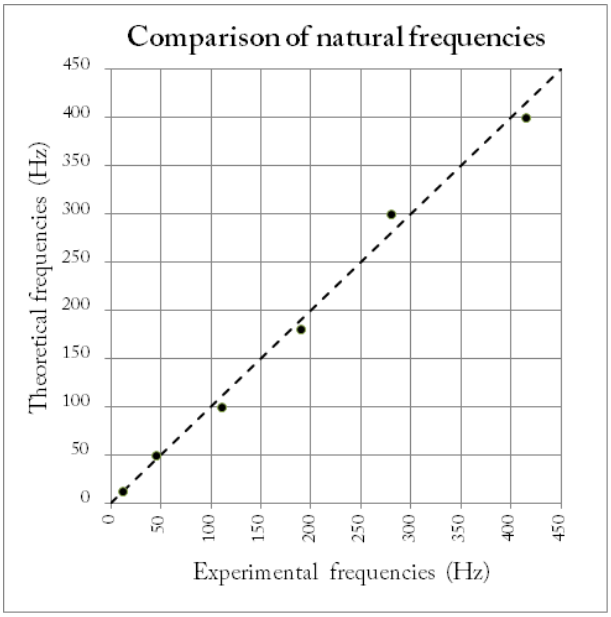

Figure 2.1. Example of a comparison of natural frequencies graphically.

Once the points of the graph have been placed, they should appear close to a straight line of slope 1, which is the dashed line in Figure 2.1. If they appear close to a line of a different slope, there may be an erroneous material property in the predictions of the numerical model. On the other hand, if the points appear spread about the line, there might be an important failure of the theoretical model in representing the structure and it should be re-evaluated. A special case is where the points diverge to some extent from the line but in a systematic and not random fashion, as this would imply that there is a particular characteristic which deviates the frequencies.

It is important to mention that the points plotted from experimental and theoretical models must be from the frequencies of the corresponding modes, as there is no guarantee that the modes match in order because the experimental model could lack of some mode. Therefore, an analysis to identify the pairs of modes is required, which can be done with the mode shape correlation method explained in the next section.

Comparison of mode shapes

As mentioned before, in order to compare frequencies, it is important to know how these frequencies are paired -one experimental with one theoretical- with information about the mode shapes. Similarly, when comparing only mode shapes

graphically -especially for complicated structures- it is difficult to differentiate and pair modes that are close with each other, and the additional information about the natural frequency is required. Therefore, it is recommended to make these comparisons of mode shapes at the same time as with the natural frequencies.

Graphically:

When using the modal shape, there is more data to handle for each mode and one way of comparison would be graphically: plotting the deformed shape for each experimental and numerical models and overlaying pairs of mode shapes. Nevertheless, one drawback is that their differences are difficult to interpret and the plots are difficult to understand, as there is to much information.

One solution (Ewins, 2000a) would be making an $x - y$ plot, where each experimental and theoretical component of the mode shape vector is plotted, similarly as explained in the frequency plot. An example is shown in Figure 2.2. The points are related to the modal coordinates and should appear close to a straight line passing through the origin. If both sets of mode shape vectors are mass-normalized, which is usually the case, the line should have a slope of 1. Again, if the points are close to a straight line with a slope different from 1, this would mean that some mode shapes are not mass-normalized or there is a error of scaling. And if the points are widely distributed about the line, this would imply that there is inaccuracy in one of the sets; if the case is excessive, it might be because the compared eigenvectors do not correspond to the same mode.

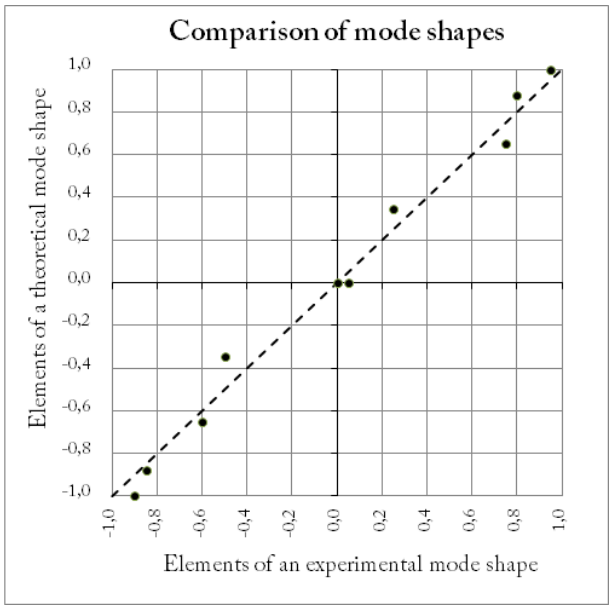

Figure 2.2. Example of a comparison of mode shapes graphically.

Numerically:

Some methods for comparing the measured and theoretical mode shapes numerically have been developed, in which some simple statistical properties are computed for the study pair of modes.

The method which calculates the slope of the best straight line through all the points, plotted in the graphical method above, is called the modal scale factor (MSF) and is defined as (Ewins, 2000c):

$$MSF\left(\phi_a, \phi_m\right) = \frac{\{\phi_m\}\{\phi_a\}^*}{\{\phi_a\}\{\phi_a\}^*} \tag{2.5}$$

Where:

- $\{\phi_a\}$ is the analytical or theoretical mode shape vector.
- $\{\phi_m\}$ is the measured mode shape vector.
- * is the conjugate operator, assuming that experimental data might be complex.

As closer to 1 the slope is, the better correlated the mode shapes should be. But it is worth to mention that the MSF parameter shows no indication of the quality of the points with respect to the straight line, it only gives its slope.

Other two parameters for comparing modal shapes numerically, MAC and COMAC, will also be explained. These will be developed in a new Section, as they require a better attention an explanation.

The Modal Assurance Criterion (MAC)

This parameter is referred as the Mode Shape Correlation Coefficient (MSCC) or more popularly known as the Modal Assurance Criterion (MAC). It is a measure of the least-squares deviation of the points from the straight line and is calculated as the normalized scalar product of the two sets of mode shape vectors:

$$MAC(r,q) = \frac{\left|\{\phi_a\}_r^T \{\phi_m\}_q\right|^2}{\{\phi_a\}_r^T \{\phi_a\}_r \{\phi_m\}_q^T \{\phi_m\}_q} \tag{2.6}$$

Here:

- $\{\phi_a\}_r$ is the analytical mode shape vector of mode r.
- $\{\phi_m\}_q$ is the measured mode shape vector of mode q.
- ^{T} is the transpose operator.

As happened with the MSF, the MAC does not indicate the degree of similarity with the straight line. Therefore, although these methods are useful to compare paired mode shape data, they should be used in conjunction with the plots explained in Section 2.3.1.

Again, a value of MAC close to 1 indicates that the two mode shapes are well-correlated and, on the other hand, a value close to 0 suggests the contrary. The difference between MSF and MAC must be noted. In the ideal case in which the two paired mode shapes are identical, both MAC and MSF have a value of 1. Nevertheless, in case they differ by a scalar multiplier γ, while $MSF = \gamma$, we have $MAC = 1$, as the to modes are still perfectly correlated.

Having a set of ϕ_m measured modes and a set of ϕ_a analytical modes, they can be computed in a table ϕ_mxϕ_a Modal Assurance Criteria and present these in a matrix which could indicate clearly which experimental mode relates to which predicted one. It is considered that values of more than 0,9 represent well-correlated modes and values of less than 0,1 as uncorrelated modes.

Some reasons why the value of MAC can be below 1, apart from having an erroneous model, can be non-linearities in the test structure, noise in the measured data, poor modal analysis of the measured data or inappropriate choice of DOFs included in the correlation. (Ewins, 2000c).

The MAC method, or some variations of it, is widely used. Nevertheless, it is important to be aware of the dangers caused by misinterpretation. Ewins (2000c) pointed out all these risks and Allemang (2003) explains the development of MAC and other related methods during 20 years. A recent article by Pastor et al. (2012) reviews the use of MAC.

The Coordinate Modal Assurance Criterion (COMAC)

The COMAC is an extension of the MAC. The degrees-of-freedom have an importance in the similarity between two pair modes, although they do not appear explicitly in the MAC coefficients. Therefore, the Coordinate MAC (COMAC) was proposed in order to obtain a quantity that expressed the dependence between the degrees-of-freedom and the similarity of the mode shapes. This is achieved in that the measure of similarity was presented as a function of the individual DOFs.

In the calculation of the MAC between two vectors, a summation is made over all the DOFs included, resulting in a single coefficient for that pair of modes. The first step in the calculation of the COMAC is to preserve the individual elements in that summation, where each refers to one particular DOF (Ewins, 2000c). Therefore, for two sets of modes that are to be compared, there will be a value of COMAC computed for each DOF. If a restriction to already paired modes is made, then the data obtained contain information about the quality of the similarity between these properly matched mode shape vectors.

In this case, once each mode pair has been identified, the COMAC for an individual degree-of-freedom j is given by (Lieven & Ewins, 1988):

$$COMAC(j) = \frac{\left(\sum_{r=1}^{L} \left|{}^{j}\phi_{ar}{}^{j}\phi_{mr}\right|\right)^2}{\sum_{r=1}^{L} \left({}^{j}\phi_{ar}\right)^2 \sum_{r=1}^{L} \left({}^{j}\phi_{mr}\right)^2} \tag{2.7}$$

In which:

- r is an individual correlated mode pair.
- L is the total mode pairs available, which can be less than the total number of modes in both sets.

Once again, a value close to 1 indicates that both modes are similar. The COMAC, unlike the MAC, has no difficulties in comparing modes which have close frequencies. However, the correct interpretation can be difficult. The experience says that systematic patterns of COMAC values normally indicate systematic sources of discrepancy between the two models, even if these are not immediately located (Ewins, 2000c).

Once the experimental data and the theoretical data are paired and compared, the procedure of updating can begin.

Chapter 3

Model Updating methods

As explained in Chapter 1, in order to solve the problem of inaccuracy, computational methods have been developed to update the Finite Element Model. During decades, some methods were and are still being proposed to solve this problem. Some of the most used, especially in model updating, can be classified according to in Figure 3.1 and will be described in this Chapter.

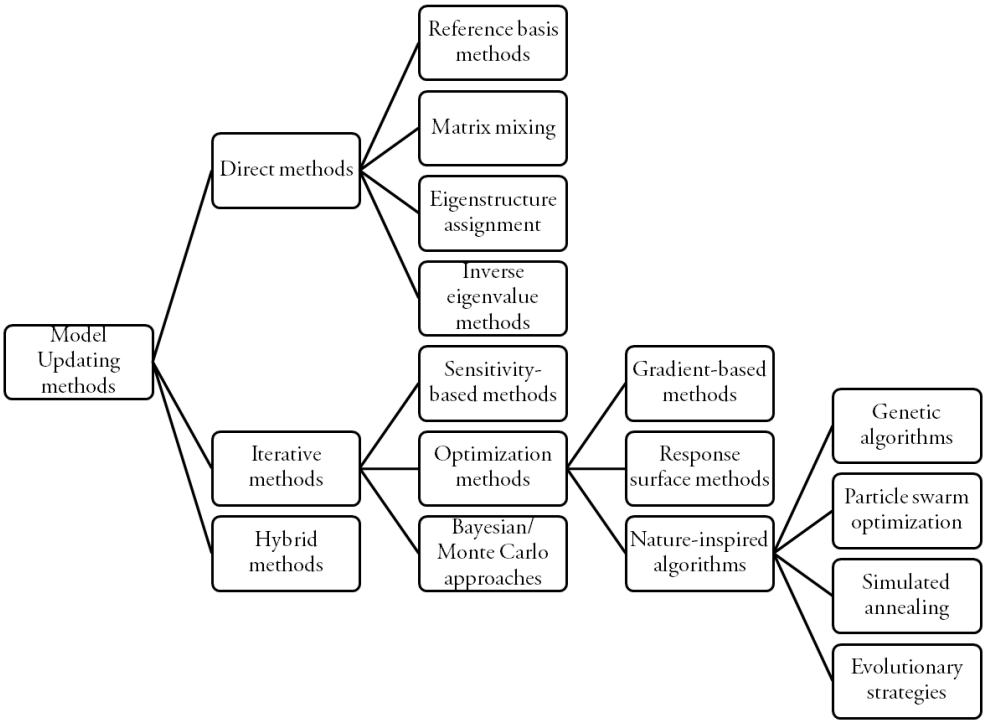

Figure 3.1. Scheme of some different types of Model Updating methods.

There are basically two distinct Finite Element Model Updating methodologies in structural dynamics: the direct methods and the iterative methods. Their ad-

vantages and disadvantages will be explicated in the following two Sections, 3.1 and 3.2, and they may be summarized as in Figure 3.2.

Type of method	Direct methods	Iterative methods
Advantages	• One-step procedure. • Exact modal properties.	• Connectivity of nodes ensured. • Physical meaning.
Disadvantages	• Lack of physical meaning. • Connectivity of nodes not ensured. • Matrices fully populated. • Reproduction of noise.	• Step-by-step procedure. • Convergence problems. • Less computational efficient. • Many parameters to update.

Figure 3.2. Table resuming the main advantages and disadvantages of both direct and iterative methods.

3.1 Direct methods: advantages and disadvantages

Direct methods directly update the elements of the stiffness and mass matrices of the Finite Element Model in a one-step procedure, which means that they have the great advantage of not requiring iteration. Furthermore, direct methods represent exactly the experimental modal properties; however, they do so without any regard to changes in physical parameters. This implies that the resulting changed matrices will have little physical meaning and will not be able to relate to physical changes in the original model. Moreover, the connectivity between nodes is not ensured and the matrices will be fully populated, and not sparse as the initial matrices of the FE model.

In addition, the advantage of the perfect matching of the updated FE model to the test data, is also weakened by the problem of reproducing unwanted noise collected in the measurements. If the updated model exactly reproduces inaccurate noise measurements, any further analysis may be faulty. Thus, in order to have accurate modelling, some requirements must be fulfilled by the direct method and high quality measurements must be gathered.

3.2 Iterative methods: advantages and disadvantages

When using iterative methods, on the other hand, the physical parameters are iteratively improved by a step-by-step approach until the Finite Element Model reproduces the experimental data to a certain degree of accuracy. Given its nature, the connectivity of nodes is assured and the mass and the stiffness matrices have physical meaning.

The iterative methods are usually related to a penalty or objective function involving the mode shape and the eigenvalue data. The penalty functions are generally non-linear functions of the parameters and are the functions which need to

be minimized. Therefore, an iterative procedure is required, with the associated convergence problems derived from it. Moreover, they require the evaluation of the analytical model at each iteration and they are less computational efficient than the direct methods. Penalty functions have a wide number of parameters to update, which could be matrix elements, structure parameters or physical quantities; and choosing a specific set is difficult.

3.3 Types of direct methods

There are basically four types of direct methods (Friswell & Mottershead, 1995): reference basis methods, matrix mixing approach, eigenstructure assignment and inverse eigenvalue methods.

3.3.1 Reference basis methods (Lagrange multiplier methods)

These methods were introduced by Baruch & Itzhack (1978) and Berman (1979), and they consider three parameters: the measured modal data, the analytical mass matrix and the analytical stiffness matrix. One of these three quantities is assumed to be exact -a reference- and, therefore, the other two are updated separately by minimizing an objective function subject to exact constraints imposed by Lagrange multipliers.

Baruch (1982) considers the modal matrix as the reference and corrects the mass matrix so that measured eigenvectors are orthogonal with respect to the mass matrix, and finally the stiffness matrix is updated. Berman & Nagy (1983) assumes the modal data to be exact and updates the analytical mass and stiffness matrices, which could be considered better since the measured data is not updated. Firstly, the mass is updated in order to be orthogonal to the measured data and, secondly, the stiffness matrix is updated. Caesar (1986) suggested other methods which updated firstly the mass matrix and then the stiffness matrix, or viceversa. And Wei (1989) proposed to update the mass and the stiffness matrices simultaneously.

3.3.2 Matrix mixing approach

This method was first introduced by Thoren (1972) and Ross (1971) and later developed by Caesar (1987) and Link et al. (1987). Usually, the number of measured modes is considerably less than the number of analytical degrees of freedom due to the lack of test data. This means that the mass and stiffness matrices cannot be constructed directly. Therefore, the matrix mixing approach expands the measured mode shapes as to have the same dimensions as the analytical degrees of freedom. In other words, they use the data from the Finite Element Model to fill in the gaps to the measured data in order to be able to calculate directly the mass and stiffness matrices. Due to that expansions, the matrices appear to be fully populated and the physical connectivity between the nodes of the structure is lost.

3.3.3 Eigenstructure assignment methods

Minas & Inman (1988) were the first ones to deal with this method, which reproduces the measured eigenvalues and eigenvectors. Given the output variables -measured- and having some input variables which are able to supply excitation to the system, the method creates a linear combination of the output variables which would give the wanted excitation signal. Therefore, the structure is forced to respond in a predetermined manner. The modifications are made to the stiffness and damping matrices, while the analytical mass matrix is not changed.

The advantage of this methods is that it requires less measured eigenvectors, which are difficult to find, and the unmeasured mode shape terms are expanded by using a well-represented Finite Element Model. Their main disadvantages are: they have a high computational cost, some of the output and input need to be specified, they lack of physical meaning and the updated matrices might not be positive semi-definite.

3.3.4 Inverse eigenvalue methods

These techniques, described by Gladwell (2004), build the mass and stiffness matrices of the vibrating structure from a knowledge of experimental data. Therefore, measured and analytical data are mixed to obtain the updated model. Thus, rather than updating the finite element model, these methods aim at determining structural modifications to be implemented on a physical system in order to match particular eigenmodes and natural frequencies.

3.4 Types of iterative methods

Three kind of iterative methods can be clearly distinguished: the sensitivity-based methods, the optimization methods and the Bayesian approaches.

3.4.1 Sensitivity-based methods

The sensitivity-based methods assume that experimental data are perturbations of the original Finite Element Model. Thus, structural modifications are done in order to change the experimental data results so that it is as close as possible to the analytical data. It is worth to mention that it only works if these modifications are small. For these methods, a number of sensitivity coefficients must be calculated, defined as the rate of change of a particular response quantity with respect to a change of a model parameter.

These methods are based on the calculation of the derivatives of the modal properties or the frequency-response functions. Many algorithms have been developed in order to calculate these derivatives. One of them was proposed by Fox & Kapoor (1968), who found exact expressions to calculate these derivatives of an undamped system for complex structures.

In the 1980s, other methods were proposed using orthogonal relations with respect to the mass and stiffness matrices. Later on, in the 1990s, Ben-Haim & Prells (1993) developed the concept of selective sensitivity for dynamical excitation in order to reduce the ill-conditioning on identification of large systems, estimating only a few parameters at a time, and Lin et al. (1995) improved the inverse eigensensitivity method in order to avoid its classical drawbacks -error assumptions and slow convergence-, in which employed both analytical and experimental modal data to calculate the eigensensitivity coefficients.

Farhat & Hemez (1993) proposed a two-step alternated procedure in which, at each iteration, the measured mode shapes are first expanded assuming that the model is error free and then the model parameters are corrected assuming that the expanded mode shapes are exact. This is implemented in an element-by-element method which is capable of detecting local errors. And Alvin (1997) modified and extended the previously-developed method in order to minimize a dynamic displacement quantity, to include Bayesian estimation concepts and to improve convergence properties by linearisation.

3.4.2 Optimization methods

As the name suggests, these methods solve an optimization problem, where the updating parameters are those in the Finite Element Model -such as material properties or geometrical dimensions- and the objective function is some measure of the distance between the finite element predicted data and measured data -such as natural frequencies or mode shapes-.

The optimization methods are frequently used for model updating, as they are flexible and can be used for many numerical models. One drawback is that, in some optimization methods, local optima can lead to problems. Moreover, it is usual to have a high number of uncertain parameters in the model; therefore, a sensitivity analysis is recommended to reduce the number of updating parameters.

Basically, three standard optimization methods are distinguished: the gradient-based methods, the response surface methods and the nature-inspired algorithms.

Gradient-based methods

The gradient-based optimization methods are deterministic methods which require the use of gradients of functions. They converge to local optima and are fast if the objective function has the right assumptions.

The most common are the descent methods. Their strategy is to iteratively search a minimum of the objective function $f(x)$, which is approximated by a terminated Taylor series expansion around the initial chosen point x_0. At each iteration i, the following process is followed:

1. Check the convergence criterion of $f(x_i)$.

2. Compute a direction d_i and a step size l_i.

3. Evaluate the new point $x_{i+1} = x_i + d_i \cdot l_i$, compute the objective function $f(x_{i+1})$ and go to step 1.

In order to compute the direction d_i, different approximations of the target function are proposed, such as the Steepest Descend Method with a linear approximation or the Newton's Method with a quadratic approximation. Moreover, the step size must be also chosen.

More advanced methods are Quasi-Newton methods, the sequential quadratic programming or the augmented Lagrangian. In Model Updating, gradient-based methods are using mainly Quasi-Newton methods. Some of the most popular are, for example, the Broyden-Fletcher-Goldfarb-Shanno method (BFGS) (Broyden, 1970; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970) or a Non-linear Programming using a Quadratic or Linear Least-square algorithm (NLPQL) (Schittkowski, 1986).

It is important to mention that the performance of a gradient-based method strongly depends on the initial values supplied. Several trials with different initial values might be necessary if there is no previous knowledge of the result. They are also restricted to small design variables, the functions and gradients must be evaluated with high precision and the problem should be smooth and well-scaled.

Response surface methods (RSM)

The response surface method was first introduced by Box & Wilson (1951) and it is an optimization method which takes into account different inputs -the parameters which will be updated- and their responses -the error between the measured data and the FE model, which is the objective function-. Its aim is to identify the combination of design variables that produces the best response. Then, the method replaces the implicit function of the original design optimization problem with a constructed approximation function, called response surface equation. This equation is normally simple and, therefore, has low computational cost, as opposed to a full FE model.

The optimization itself, then, is performed on the response surface using other optimization techniques such as gradient-based or nature-inspired algorithms. This means that the RSM consists of, firstly, the response surface approximation equation and, secondly, the optimization procedure. Many techniques have been used to approximate the response, such as least square or moving least square approximations, polynomial approximation or neural networks. Nevertheless, the objective is to have a well-qualified response surface to represent global trends of the optimization problem.

The main steps in the process of the RSM include the following (Ren & Chen, 2010):

- Selecting the structural parameters and defining the level for each selected parameters.

- Calculating the response features in design space.

- Performing the final regression followed by a regression error analysis to create the response surface model of the structure.
- Measuring the response features of the structure and constructing the corresponding objective functions to be minimized.
- Carrying out the Finite Element Model Updating within the established response surface model. The updated parameters are obtained and transferred to the original finite element model.

The extremely fast convergence is the main advantage of this method, but it should be applied to reasonably smooth problems with not more than 10 continuous variables.

Ren & Chen (2010) did a recent work in a RSM, as well as Marwala (2010).

Nature-inspired algorithms

The nature-inspired algorithms are either evolutionary algorithms or related search heuristics algorithms, both stochastic search methods. As the name indicates, their inspiration comes from the nature: they are based on and imitate natural biological evolution processes such as adaptation, selection and variation. A sample of artificial individuals is initially generated in the space of possible solutions which is based on the Darwin principle "survival of the fittest", where the evolution process is the updating process.

As gradient-based information may be not available, nature-inspired algorithms are very useful because they do not need it. Moreover, these algorithms are better in finding the global minimum, instead of the local minimum. Nevertheless, the main drawback of these methods is their high computational cost in order to obtain the same accuracy of other methods. Nature-inspired algorithms are recommended in cases when gradient-based optimizations or response surface methods fail, when there are discrete variables dominating the response, as well as when there is a high number of variables and/or constraints.

According to Zang et al. (2010), the important stages involving efficient algorithms are: observing and summarizing the behaviour of the creatures in the nature, establishing a raw model to represent the behaviour, converting it to mathematical module -with assumptions and initial parameters-, developing the code to simulate the behaviour, testing the algorithm theoretically and experimentally, and refining the parameters setting to achieve better performance.

Some of the most popular nature-inspired algorithms are the Genetic Algorithms (GA), the Evolutionary Strategies (ES), the Particle Swarm Optimization (PSO), the Simulated Annealing (SA) and the Neural Networks (NN).

1. **Genetic Algorithms (GA)**

   This type of nature-inspired algorithms was created by Holland (1992), who pointed that they can solve complex problems that even their creator do not fully understand.

The Genetic Algorithms approach the solution iteratively as a competition among a population of evolving candidate problem solutions. The objective function -the error between the experimental data and the analytical data- is evaluated for each individual in order to decide if it will contribute for the next generation of solutions, which will also involve reproduction and mutation.

The process of the GA, according to Marwala (2010), is the following:

- *Initialization.* A random sample of solutions is generated to form the initial population, covering the entire range of possible solutions.
- *Crossover.* The genetic information -chromosomes- of the solutions is mixed -cut and exchanged-.
- *Mutation.* At some point, the chromosomes are inverted randomly in order to introduce new information and avoid local optimum solutions.
- *Selection.* A proportion of the existing population is chosen to breed a new population. The solutions that are closer to the optimum have higher probability of being selected.
- *Termination.* Crossover, mutation and selection -called generation- is repeated until a termination condition has been achieved, which can be that a solution satisfies the objective function, the maximum number of iterations has been reached or the solution has converged.

The main disadvantage of these type of algorithms is that it is very slow, which makes it hard to implement to a big structure. The advantages are that it is easier to find the global minimum and that they do not need the gradient of the objective function.

The Genetic Algorithms are very popular in Finite Element Model Updating of structures, some of the most recent are implemented by Tu & Lu (2008), Marwala (2010) and Ribeiro et al. (2012).

2. **Evolution Strategies (ES)**

Evolution Strategies are quite similar to GA and were first proposed by Rechenberg in the 1960s (Rechenberg, 1965) and further developed by Schwefel (Schwefel, 1975). In contrast to GA, ES were designed from the beginning as optimisation methods.

They are based on the concept of evolution inside evolution. Each individual is represented by its genetic code and a set of strategy parameters that model its behaviour in the environment. In this case, evolution means changing both the genetic and strategy parameters. In addition, changes due to mutation are only accepted if they make an improvement of the solution -otherwise they are discarded- and crossover can also be produced from more than two solutions.

Some recent articles concerning Evolutionary Strategies were written by Perera & Ruiz (2008) and Jafarkhani & Masri (2011).

3. **Particle Swarm Optimization (PSO)**

   The Particle Swarm Optimization method is a stochastic, population-based approach that was inspired by a mathematical description of the swarming of birds. Swarm intelligence allows them to communicate and interact with other individuals in order to undertake problems and develop a similar way of facing them. Therefore, PSO takes into consideration two factors: group knowledge and individual knowledge. Each individual of the swarm acts by balancing between its individual knowledge and group knowledge (Marwala, 2010).

   As with the other optimization methods, an objective function -the difference between experimental and analytical data- is defined in order to find the optimum. A social network representing the population of possible solutions is firstly created, in which each of the individuals will have neighbours to interact with. Then the updating process starts.

   Each particle -possible solution- is evaluated in the objective function at each position of the iteration and can remember the location where it had its best result. This best location is shared to the neighbours, who will move along the space having in consideration its local best and the ones from their neighbours, until convergence and the optimum solution are achieved.

   The advantages of these methods are (Perez & Behdinan, 2007) that the scaling of the design variables -experimental data- is well-done, it is easy to execute and implement, it does not require the calculation of the derivatives of the objective function, it contains a small amount of adjustable parameters and it successfully obtains a global solution. The disadvantage of the PSO is that it might find a local solution and, thus, could converge prematurely.

   Recent works in PSO in structures include Marwala (2010) or Mthembu et al. (2011a).

4. **Simulated Annealing (SA)**

   Simulated Annealing is essentially a Monte Carlo technique used to identify a global solution, which mimics the annealing process. Annealing refers to the recrystallization process of a liquid or solid when cooled down, and the analysis of the behaviour of substances as they cool.

   In this process, firstly the object is heated until it is molten and secondly its temperature is slowly decreased until the object is in thermodynamic equilibrium. As temperature diminishes, the mobility of the molecules reduces and their tendency is to align themselves in crystalline structure, reaching the minimum-energy state. In order to ensure this alignment and not to have an

amorphous state -a system trapped in a local minimum-energy state-, cooling must occur at a sufficiently slow rate.

In the case of Simulated Annealing, an algorithmic implementation of the annealing process to find the optimum of the objective function is created, where the minimum of the objective function represents the minimum energy of the system. SA uses a random search strategy which, obviously, accepts new positions that minimize the objective function, but also keeps some changes that are not ideal -which increase it-.

It was Levin & Lieven (1998) who introduced the Simulated Annealing algorithms in Finite Element Model Updating.

Their advantages (Aarts & Korst, 1997) are that it can handle objective functions with many design variables, a global optimum solution is assured, it is simple to code and it offers a sufficient optimal solution which can be used in practice.

It also has some disadvantages (Salamon et al., 2002), for instance, it can be extremely slow if the objective function is expensive to compute, if there are no local minima it is not that efficient compared to other methods, it is highly dependent on the nature of the problem, it is difficult to identify when the optimal solution has been reached and it is also difficult to decide the cooling schedule to implement -the rate at which the temperature decreases-.

Some recent studies related to Simulated Annealing are Genovese et al. (2005), Sonmez (2007) and Martí & González-Vidosa (2010).

3.4.3 Bayesian/Monte Carlo approaches

A Bayesian approach is a procedure based on Bayes' Theorem and functions for conducting statistical inference through using the evidence -observation- to update the probability that a hypothesis may be true (Marwala, 2010). Mthembu et al. (2011b) applies a Bayesian approach in selecting the best from a set of FE models and Zheng et al. (2009) also applied a Bayesian approach for FEMU of a long-span, steel sky-bridge. Zhang et al. (2011) presents a comprehensive Bayesian approach for structural model updating which accounts for errors of different kinds.

These approaches are beyond the scope of this Master Thesis, but more information can be found in the book by Yuen (2010). This book explains the aim of the Bayesian approaches and their applications in Model Updating.

3.5 Hybrid methods

In order to combine the efficiency of local and global methods, some hybrid methods have been designed and studied. For example, Marwala (2010) explains some hybrid combinations such as Hybrid Particle-swarm Optimization and the Nelder-Mead Simplex. Jung & Kim (2011) tried an hybrid Genetic Algorithm by combining

Genetic Algorithm and the modified Nelder-Mead Simplex method. Feng et al. (2006) proposed a combination of the Genetic Algorithm and Simulated Annealing to enhance the FE model updating for two rotors.

Chapter 4

Recent bee-based algorithms

As mentioned in Section 3.4.2, many algorithms based on the nature such as GA or PSO have been developed in order to solve optimization problems. Among all nature-inspired algorithms, recent swarm intelligence algorithms are emerging and some of them are based on the behaviour of bees.

In the present Chapter some recent optimization algorithms based on the foraging behaviour of bees when looking for nectar will be explained.

4.1 The behaviour of bees

Before explaining some of the bee-based algorithms it is important to understand how they behave and perform their actions.

A colony of bees can extend itself over long distances -more than 10 km- and in multiple directions simultaneously in order to exploit a large number of food sources. It is a dynamical system which gathers information from the environment and adjusts it behaviour in accordance to it. Therefore, their procedure is expanding the seeker bees -foragers- to different fields with the objective that, in the end, flower patches with better quantities of nectar or pollen that can be collected with less effort are visited by more bees.

4.1.1 Types of foraging bees

The bee system consists of two essential components: the food sources and the foragers. The value of a food source depends on different parameters such as its proximity to the hive, the richness of energy that they provide and the ease in which this energy can be extracted. The foragers can be divided in three types: unemployed, employed and experienced foragers.

Unemployed foragers:

The unemployed foragers are supposed to have no initial knowledge about where to find the food sources. There can be two kinds of unemployed foragers:

- Scout bees (S in Figure 4.1): they do a spontaneous search. According to Seeley (1995), the amount of scout bees can vary from 5 to 30% of the total amount of bees of the hive and, on average, they represent the 10%.
- Recruit or onlooker bees (R in Figure 4.1): they look for a waggle dance from another bee, which includes the information about a good source of food and will be explained further on in Section 4.1.2.

Employed foragers:

A recruit bee becomes an employed forager (labeled EF in Figure 4.1). when it finds and memorizes the location of the food source in order to exploit it. Once on the food source, they firstly load a portion of nectar from it and then they return to the hive to unload the nectar. At this point, the employed bee has three options depending on the quality and/or quantity of the food source:

- If the nectar has decreased into a low level or it is exhausted, the bee abandons the food source and becomes an unemployed forager.
- If there is still enough amount of food source, it can continue to forage without sharing the source location.
- Or it can do the waggle dance in order to inform other bees about the food source.

Experienced foragers:

This kind of bees use their previous knowledge and memories for the location and quality of food sources. They can become:

- Inspectors: they control the recent status of food source already discovered.
- Reactivated foragers (RF in Figure 4.1): which use the information from the waggle dance. They explore the same food source discovered by themselves if other bees confirm its quality.
- Scout bees (ES in Figure 4.1): which search new patches if the food source is finished.
- Recruit bees (ER in Figure 4.1): which search for food declared through the waggle dance by other employed bees.

4.1.2 How bee colonies behave

In the previous Section, the different kind of honey bees were described. In this Section their way of communicating to each other in order to find the best food sources is explained. A graphical representation of their behaviour can be found in Figure 4.1.

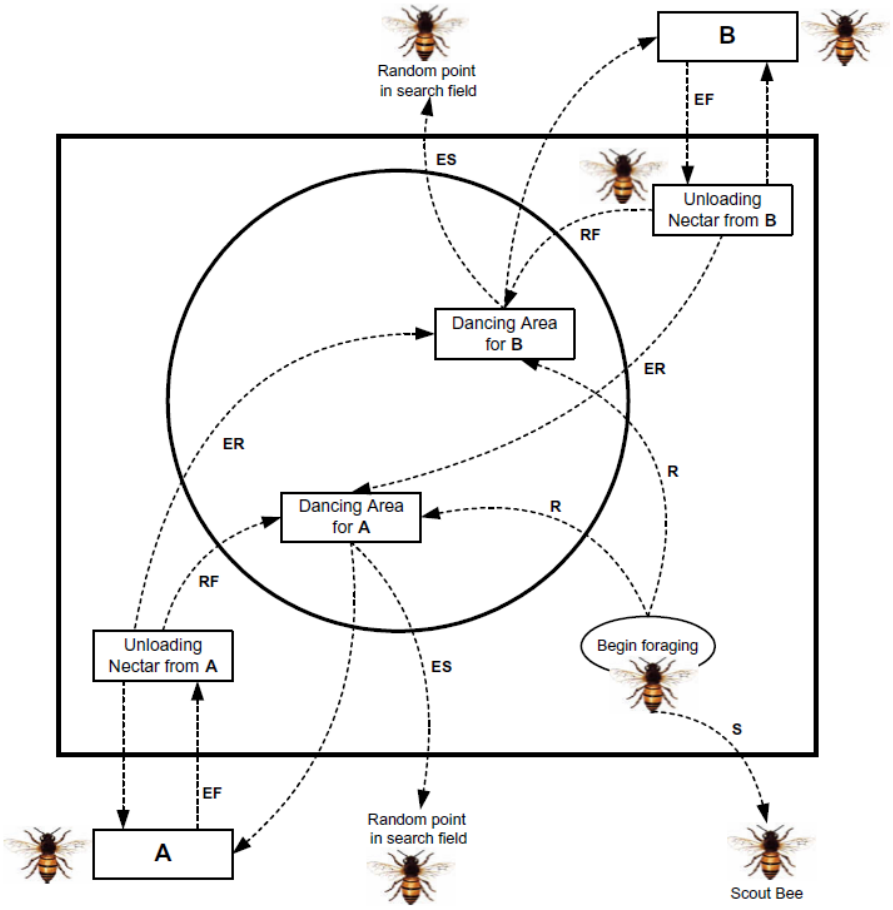

Figure 4.1. Typical behaviour of honey bee foraging (Baykasoglu et al., 2007).

Firstly, scout bees are sent to search for flower patches. They move randomly from one patch to another without any previous knowledge about them. After finding some source, they return to the hive to deposit the nectar, which is rated depending on the constituents they provide such as sugar.

The scout bees that found a patch which is rated above a certain quality threshold go to the "dance floor" and perform the "waggle dance". This dance is essential for the communication of the colony and provides information about the patch: the direction of the bee indicates the direction of the food in relation to the sun -which is adjusted in time-, the intensity of the waggle shows the distance from the hive to the food source, and the duration of the dance is related to the amount of nectar and quality of rating -also known as fitness-.

The colony evaluates the value of different patches informed by different waggle dances: they make a balance of both the quality of the food and the amount of energy needed to harvest it. Therefore, this information provided by the waggle

dance allows other bees to fly to the best food sources precisely. It must be noted that the bees' individual knowledge of the outside environment is only provided by the waggle dance.

After the waggle dance, the dancer -scout bee- goes back to the same source followed by other bees -recruit bees- that were waiting inside the hive. While there is still enough nectar in the patch which is being harvested, it will still be announced with the waggle dance and more bees will fly there.

4.2 Different optimization algorithms based on bees

Since 1990s, algorithms based on the behaviour of bees have been developed. Especially, it is during the last decade when bee swarm intelligence has inspired researchers to create new algorithms. Baykasoglu et al. (2007) made a detailed review of all bee-based algorithms and Karaboga & Akay (2009c) made a survey of bee swarm algorithms as well as their applications.

Nevertheless, to the author's best knowledge, there are only three numerical function optimization algorithms in the literature based on intelligent foraging behaviours of bee swarms: the Virtual Bee Algorithm (VBA), the Bees Algorithm (BA) and the Artificial Bee Colony (ABC).

The first one was introduced by Yang (2005), who developed an algorithm to optimize a function with two parameters in which a swarm of virtual bees is generated and start to move randomly in the space. Once they find nectar -a possible value for the function- they interact and the solution for the problem is obtained from the intensity of bee interactions.

For optimizing multi-variable functions, the other two algorithms were developed almost at the same time and both will be deeply described in the following sections.

4.2.1 The Bees Algorithm (BA)

The Bees Algorithm was first introduced by Pham et al. (2005) and further developed by him in Pham et al. (2006), Pham et al. (2007) and (Pham & Ghanbarzadeh, 2007), and its applications are reviewed in Karaboga & Akay (2009c). The procedure of this algorithm is summarized below and a pseudo-code for it is found in Figure 4.2.

This algorithm requires certain parameters to be set:

- Number of scout bees (n): it is the total number of random solutions.
- Number of selected sites (s): it is the number of solutions which produce a minimum value of the optimization function -a higher fitness value- out of the n.
- Number of elite sites (e): it is the number of solutions which produce the highest fitness values out of s.

1. Initialise population with random solutions.
2. Evaluate fitness of the population.
3. **While** (stopping criterion not met)
   //Forming new population.
4. Select sites for neighbourhood search.
5. Recruit bees for selected sites (more bees for best e sites) and evaluate fitnesses.
6. Select the fittest bee from each patch.
7. Assign remaining bees to search randomly and evaluate their fitnesses.
8. End **While**.

Figure 4.2. Pseudo-code for the BA in its simplest form (Pham et al., 2006).

- Number of bees recruited for each selected site (nsp): it is the number of neighbourhood searches around s.
- Number of bees recruited for each elite site (nep): it is the number of neighbourhood searches around e, where $nep > nsp$.
- Initial of the patches (ngh): it is the radius inside which the neighbourhood search is produced.

The process starts in step 1 with n random solutions distributed in the search space, which will be evaluated in step 2. In step 3, the iteration begins until a stopping criterion is met. Then, s and e are selected, the neighbourhood searches are performed and their fitnesses evaluated in steps 4 and 5. In step 6, the solution with the fittest value from each neighbourhood search is selected. Finally, the remaining $n - s$ solutions are selected randomly from the search space in step 7, in order to search potential solutions and avoid local optima. Therefore, the new population will be formed from representatives from each neighbourhood solution and random solutions. When the iteration finishes, the best of the optimum values obtained in the last population of solutions is expected to be the global optimum.

In an advanced BA, a shrinking constant k is defined, which will decrease ngh at each iteration if there is no progress in finding a better solution. This step would be done between steps 4 and 5, and it is called the "shrinking method".

Figure 4.3 shows the flowchart of the BA (Moradi et al., 2010).

The random solutions x_{rand} are calculated using:

$$x_{rand} = x_{min} + \alpha \cdot (x_{max} - x_{min}) \tag{4.1}$$

Where:

- α is a random vector which its elements are between 0 and 1.
- x_{min} and x_{max} are the lower and upper bounds for the solution vector, respectively.

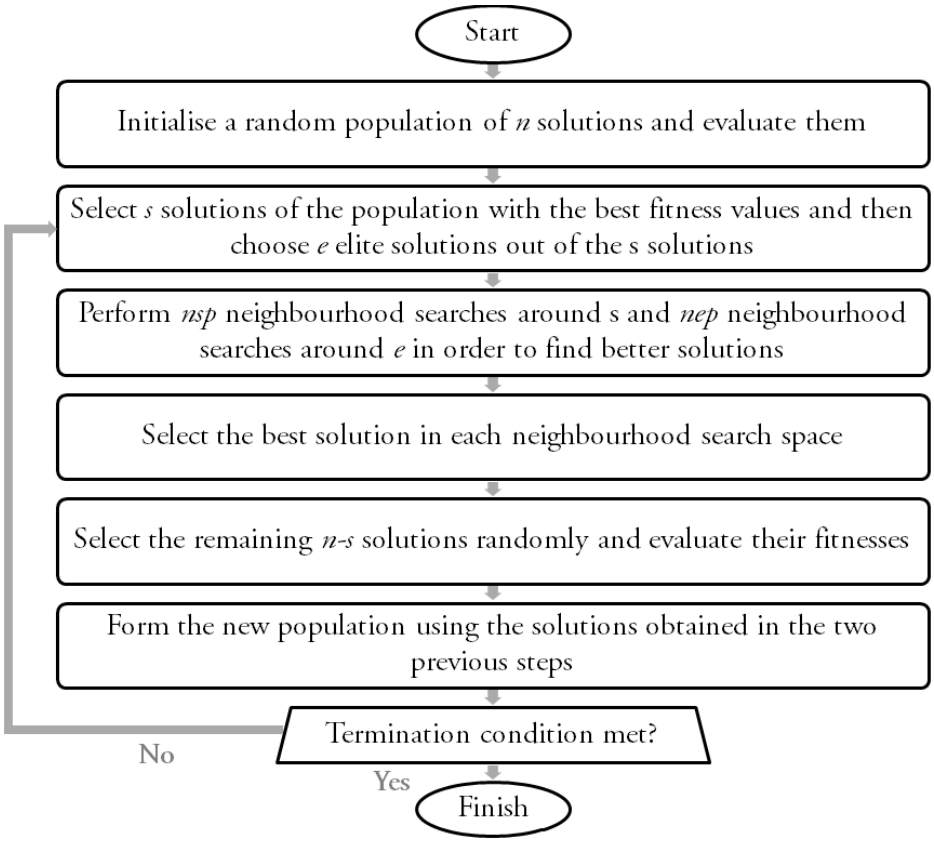

Figure 4.3. Flowchart of the Bees Algorithm.

The neighbourhood search around each element of the solution vector x_i is performed using:

$$x_{pi} = (x_i - ngh) + 2 \cdot \alpha_i \cdot ngh \tag{4.2}$$

Here:

- x_{pi} is the i^{th} element of the new solution vector obtained from a neighbourhood search around x_i with its radius equal to ngh.

The main advantage of the this algorithm is that it does not require gradient information and, therefore, it can escape from local optima (Pham et al., 2005).

4.2.2 The Artificial Bee Colony (ABC)

The Artificial Bee Colony was first introduced by Karaboga (2005) and further developed by him in Karaboga & Basturk (2007) and Karaboga & Basturk (2008), who also compared the ABC to the BA in Karaboga & Akay (2009a) and to other nature-inspired algorithms in Karaboga & Akay (2009b). Its applications are also reviewed in Karaboga & Akay (2009c).

In this model, three groups of bees are considered for the colony of artificial bees (possible solutions): employed bees, onlooker bees and scout bees, the three of them explained in Section 4.1. The first group, as well as the second, represents half of the colony and is equal to the number of food sources around the hive SN. The group of scout bees are employed bees whose food source have been exhausted. The pseudo-code for it is found in Figure 4.4.

1. Initialise population of random solutions.
2. Evaluate the population.
3. **While** (stopping criterion not met)
4. Place the employed bees on their food sources.
5. Place the onlooker bees on the food sources depending on their nectar amounts.
6. Send the scout bees to the search area for discovering new food sources.
7. Memorise the best food source found so far.
8. End **While**.

Figure 4.4. Pseudo-code for the ABC in its simplest form (Karaboga & Basturk, 2007).

In step 1 there is an initialization process in which SN random solutions are generated. In step 2 these solutions are evaluated and in step 3 the population of solutions is subjected to repeated iterations, each of them consisting of three phases. Phase 1 is performed in step 4, where each employer bee is moved onto its last memorized food source, produces a modification on this solution, evaluates both fitnesses and memorizes the best one for the next iteration. Phase 2 is accomplished in step 5, where each onlooker bee is moved onto a food source from the employed bees using a probability based selection process -as the nectar amount of a food source increases, the probability value with which the food source is preferred by onlookers also increases-, modifies it, evaluates both fitnesses and memorizes the best one. Step 6 represents phase 3, in which the abandoned solution is determined -if exists- and it is replaced with a new randomly produced solution.

Figure 4.5 shows the flowchart of the ABC (Karaboga & Akay, 2009b).

As in the Bees Algorithm, the random solutions are generated according to Equation 4.3:

$$x_{ij} = x_j^{min} + \left(x_j^{max} - x_j^{min}\right) \cdot rand \tag{4.3}$$

Where:

- x_{ij} is an initially random solution, being $i = 1, ..., SN$ the food source number and $j = 1, ..., D$ the dimension of the problem.
- x_j^{min} is the lower limit of the solution.

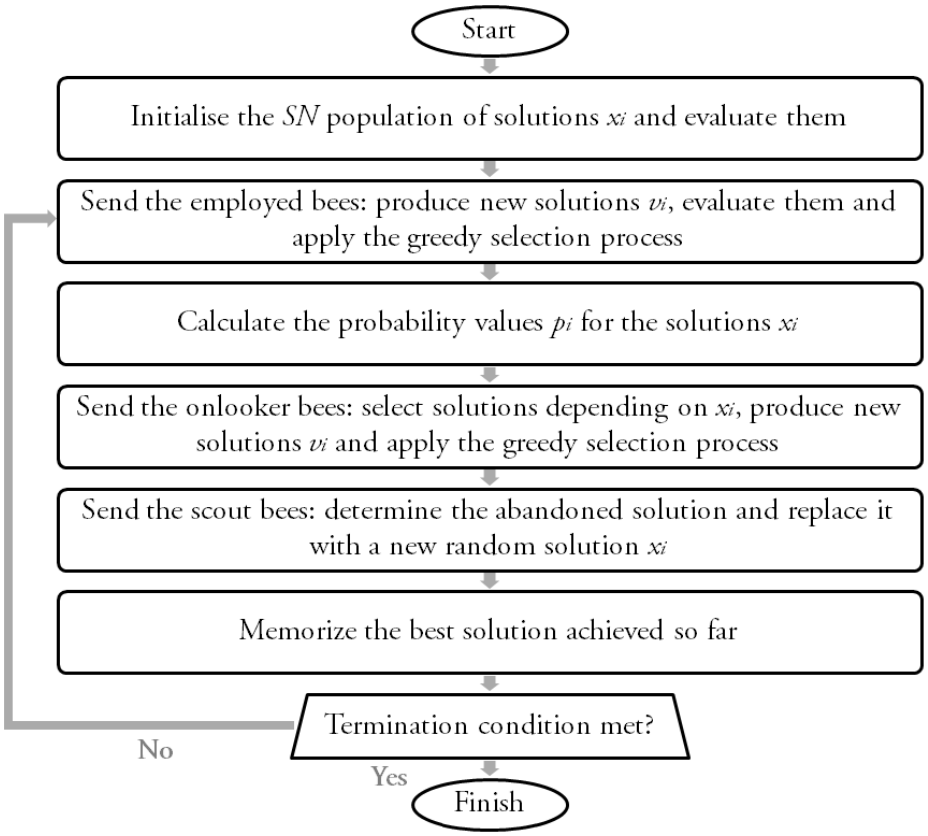

Figure 4.5. Flowchart of the Artificial Bee Colony algorithm.

- x_j^{max} is the upper limit of the solution.
- $rand$ is a uniformly distributed real random number $\epsilon[0, 1]$.

The modifications done in steps 4 and 5 in order to obtain new solutions are generated using Equation 4.4:

$$v_{ij} = x_{ij} + \varphi_{ij} \cdot (x_{ij} - x_{kj}) \tag{4.4}$$

Where:

- v_{ij} is the modified solution.
- φ_{ij} is a uniformly distributed real random number $\epsilon[-1, 1]$.
- k is an index of the solution chosen randomly from the colony: $k = fix(rand * SN) + 1$.

If a solution produced by this Equation exceeds its predetermined limit, the parameter can be set to an acceptable value. In this work, the value of the parameter exceeding its limit is set to its limit value.

The probability of an onlooker bee to choose a food source from the employer bees is calculated using Equation 4.5:

$$p_i = \frac{fit_i}{\sum_{j=1}^{SN} fit_j} \tag{4.5}$$

Where:

- p_i is the probability of solution x_i to be chosen from the onlookers.
- fit_i is the fitness of the solution x_i

When a modification is done, a greedy selection is performed in which, if the fitness of the new source is higher that the one from the previous one, the bee memorizes the new position and forgets the old one. Otherwise, it keeps the position from the old one.

A food source is abandoned if it cannot be improved further through a predetermined number of iterations -called *limit*- and it is replaced by a new randomly generated solution using Equation 4.3. In the basic ABC algorithm, only one source can be abandoned at each iteration, but in a more advanced ABC algorithm more sources can be abandoned at a time.

This algorithm requires a few control parameters to be set:

- Total number of explored food sources or solutions, which is equal to the number of employed bees and also to the number of onlookers, SN.
- Maximum number of iterations or maximum cycle number, MCN.
- Limit control parameter, this is, the maximum number of iterations that a solution is kept before it is abandoned and replaced by a new random solution, $limit$.

According to what has been explained in this Section, ABC algorithm employs four different selection processes (Karaboga & Akay, 2009b):

1. A global selection process, performed by the onlooker bees when choosing solutions from the employer bees according to their probabilities.
2. A local selection process carried out by the employed bees and the onlookers when modifying locally a source in their memory.
3. A local selection process called greedy selection performed by all bees in order to choose between the old source and the modified source according to its fitness.
4. A random selection process carried out by scout bees.

Therefore, exploration and exploitation processes are carried out together: while onlookers and employed bees carry out the exploitation process in the search space, the scouts control the exploration process. In conclusion, Artificial Bee Colony algorithm is a simple, flexible and robust algorithm that can be used for solving unimodal and multi-modal numerical optimization problems (Karaboga, 2005).

4.3 Performance of the BA and the ABC. Previous studies

Many articles have been written regarding the Bees Algorithm and the Artificial Bee Colony algorithm, as shown in the previous section about these two algorithms. Some of them deeply explain the algorithms and give results for its implementation, and some of them also compare these results with the ones obtained with other algorithms.

Their performance, therefore, has been proved to be successful, even-though the BA has been only tried once in Model Updating (Moradi et al., 2010) and ABC has never been tried and implemented in Model Updating. Thus, these two algorithms are good candidates to be implemented in Finite Element Model Updating for the present Master Thesis.

Chapter 5

Technical description of the programs

In this Chapter, a description of all the programming part is made. The steps that have been carried out are:

- Programming the Finite Element Model in order to obtain the numerical frequencies, and transforming it to a function which given certain characteristics calculates the frequencies.
- Deciding the objective function.
- Programming the Bees Algorithm.
- Programming the Artificial Bee Colony.

Both BA and ABC algorithms call the frequency function in order to compare the frequencies from the numerical model to the experimental frequencies. Moreover, ABC requires another function -called *onlookers*- to be set in order to calculate the probability of each possible solution in order to be chosen by the onlookers.

5.1 Finite Element Model

5.1.1 Basic theory

As explained in Section 2.1.1, the Equation of Motion (Equation 2.3) needs to be solved in order to obtain the modal properties of the beam. Therefore, the stiffness and mass matrix must be calculated.

In order to obtain the mass and the stiffness matrices, the theory of the 2D Euler-Bernoulli beam is considered in the present Master Thesis. No details will be presented on how these matrices are obtained, as it would be too extensive. For more information, Gavin (2012) describes the formulation of stiffness and mass matrices for structural elements.

The 2D Euler-Bernoulli beam theory's major assumption is that 'plane sections remain plane'. Therefore, no shear deformation is accounted.

According to the theory of Finite Element Modelling, the assumed 2D beam is divided in elements. Each element has two nodes, each of them with three degrees of freedom (DOF): two translational DOF -horizontal and vertical- and one rotational DOF, as shown in Figure 5.1.

Figure 5.1. Representation of a 2D Euler-Bernoulli beam element.

Considering this 2D Euler-Bernoulli beam element, which's cross section is constant and has area A and density ρ along the length of the element Le, the consistent element mass matrix is:

$$Mec = \frac{\rho ALe}{420} \cdot \begin{bmatrix} 140 & 0 & 0 & 70 & 0 & 0 \\ 0 & 156 & -22Le & 0 & 54 & 13Le \\ 0 & -22Le & 4Le^2 & 0 & -13Le & -3Le^2 \\ 140 & 0 & 0 & 70 & 0 & 0 \\ 0 & 54 & -13Le & 0 & 156 & 22Le \\ 0 & 13Le & -3Le^2 & 0 & 22Le & 4Le^2 \end{bmatrix}$$

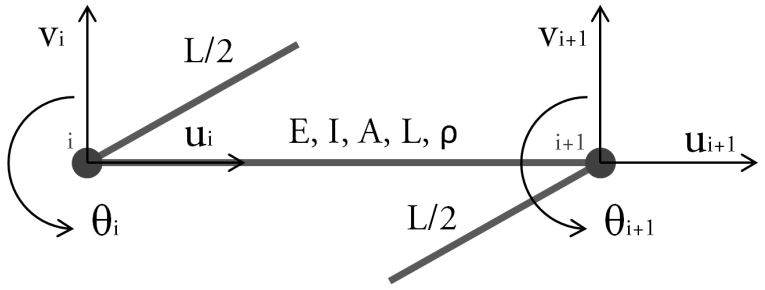

Figure 5.2. Representation of a 2D Euler-Bernoulli beam for the lumped mass matrix.

Nevertheless, the consistent mass matrix tends to overestimate the natural frequencies of the beam. Moreover, it is normally a full matrix. In order to solve these problems, a lumped mass matrix is created. For the 2D Euler-Bernoulli beam, each matrix element is obtained by assigning half the element mass to each of the translational degrees of freedom and the moment of inertia of half the beam element

about the respective nodes to the rotational degrees of freedom, as in Figure 5.2. The lumped element mass matrix obtained is:

$$Mel = \frac{\rho ALe}{2}. \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & Le^2/12 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & Le^2/12 \end{bmatrix}$$

The lumped mass matrix is a diagonal, simpler matrix. On the other hand, it tends to underestimate the natural frequencies of the beam. In order to palliate the errors from both mass matrix formulations, a combination of these two matrices is carried out:

$$Me = (1 - \beta)Mel + \beta Mec$$

Where $\beta \epsilon [0, 1]$. If $\beta = 0,5$ the mass matrix is called average mass matrix.

In order to obtain the element stiffness matrix, two effects must be considered separately: extension effects and bending effects. The formulation of the first one involves the area of the section and the second one the moment of inertia. For homogeneous isotropic beams, which are the ones considered in this Master Thesis, the element stiffness matrix obtained is:

$$Ke = \begin{bmatrix} AE/Le & 0 & 0 & AE/Le & 0 & 0 \\ 0 & 12EI/Le^3 & -6EI/Le^2 & 0 & -12EI/Le^3 & -6EI/Le^2 \\ 0 & -6EI/Le^2 & 4EI/Le & 0 & 6EI/Le^2 & 2EI/Le \\ AE/Le & 0 & 0 & AE/Le & 0 & 0 \\ 0 & -12EI/Le^3 & 6LeEI/Le^3 & 0 & 12EI/Le^3 & 6EI/Le^2 \\ 0 & -6EI/Le^2 & 2EI/Le & 0 & 6EI/Le^2 & 4EI/Le \end{bmatrix}$$

Where E is the Young's Modulus and I is the bending moment of inertia of the cross section, calculated as follows:

$$I = \frac{1}{12} WT^3 \tag{5.1}$$

Here, W is the width of the beam and T the thickness of it.

Once the element stiffness and mass matrices are obtained, the stiffness and mass matrices of the beam are obtained by an assembly of the element matrices. Then, the boundary conditions of the structure are imposed. Finally, the eigenvalues -frequencies- and eigenvectors -mode shapes- are calculated. More details and explanations can be found in Chopra (2001).

5.1.2 Parameters of the FEM

The initial parameters chosen for the numerical model (experimented cantilever beam, see Chapter 6) can be found in Table 5.1.

Parameter	**Notation**	**Value**
Number of nodes (−)	No	100
Length (m)	L	$1,3$
Width (m)	W	$0,06$
Thickness (m)	T	$0,004$
Young modulus (N/m^2)	E	$2,100 \cdot 10^{11}$
Density (kg/m^3)	ρ	$7,850 \cdot 10^3$
Average parameter (−)	β	$0,5$
Horizontal stiffness in first node (N/m)	$kh1$	10^{10}
Vertical stiffness in first node (N/m)	$kv1$	10^{10}
Rotational stiffness in first node (N · m/rad)	$kO1$	10^{10}
Horizontal stiffness in last node (N/m)	$kh2$	0
Vertical stiffness in last node (N/m)	$kv2$	0
Rotational stiffness in last node (N · m/rad)	$kO2$	0

Table 5.1. Initial parameters to be set.

The dimensions of the beam are taken from the experimental beam and its properties are assumed as the usual parameters of steel. The stiffness of the first and last nodes are the ones from a cantilever beam, as it is be the type of beam to analyse. A number of 10^{10} has been initially chosen for the stiffness of the fixed node as an approximation of infinite -totally fixed node-.

A convergence study was performed in order to choose the number of nodes on which the beam is to be divided. The relation between time and accuracy was considered to choose the right number of elements. Table 5.2 shows the obtained five first frequencies, the time consumed and the error -which is the objective function from Equation 5.3, explained later on- compared to the exact frequencies.

Elements	Frequencies (Hz)					Time (s)	Error
9	1,9696	12,1734	33,5128	64,2901	103,6988	1,10	$1,05 \cdot 10^{-2}$
24	1,9764	12,3616	34,5265	67,4392	111,0465	1,25	$2,43 \cdot 10^{-4}$
49	1,9777	12,3855	34,6590	67,8644	112,7780	1,26	$1,38 \cdot 10^{-5}$
99	1,9775	12,3913	34,6908	67,9669	112,3278	1,70	$8,64 \cdot 10^{-7}$
199	1,9775	12,3927	34,6985	67,9920	112,3890	7,20	$2,13 \cdot 10^{-7}$
499	1,9775	12,3931	34,7006	67,9989	112,4060	115,00	$2,07 \cdot 10^{-7}$

Table 5.2. Frequencies, time and error of the FEM with different number of elements.

Table 5.3 shows how the mode shapes vary depending on the number of elements in which the FEM is divided into. The smoothness of the mode shapes does not seem to be visibly improved from 49 elements.

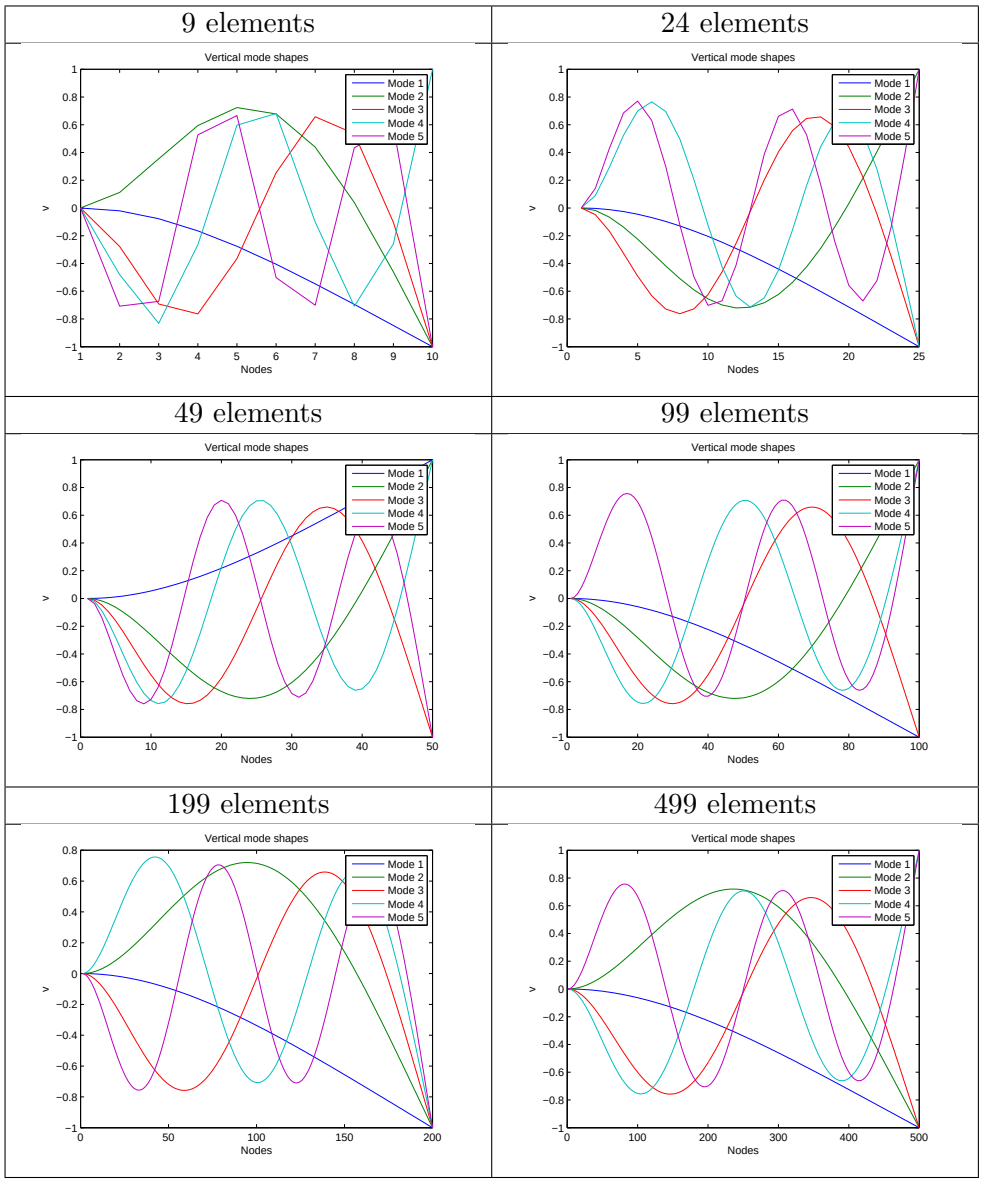

Table 5.3. Mode shapes of the FEM with different number of elements.

In order to compare objectively which is the appropriate number of elements, some calculations regarding the relation between the number of elements, the error and the time consumed between the results from one FEM and the previous one is considered, which is shown in Table 5.4. It can be seen that until 99 elements, the

time consumed does not increase and more accuracy is gained. Between 99 and 199 elements, there seems to be a good improvement; nevertheless, this is achieved to the detriment of a huge increase of time.

Elements	$Elem_1/Elem_{i-1}$	$Error_i/Error_{i-1}$	T_i/T_{i-1}
9	-	-	-
24	0,38	42,5145	0,88
49	0,49	14,8086	0,99
99	0,49	6,0278	0,74
199	0,50	57,0696	0,24
499	0,40	15,8153	0,06

Table 5.4. Comparison of error frequencies and time of the FEM with different number of elements.

Therefore, for all the reasons exposed, the final number of elements chosen is 99, as it provides good accuracy at a reasonable speed.

With the conditions from the previous Section, the obtained mode shapes are the ones in Figure 5.3 and the frequencies are:

$$f = [1,9775 \quad 12,3913 \quad 34,6908 \quad 67,9669 \quad 112,3278] \text{ Hz}$$

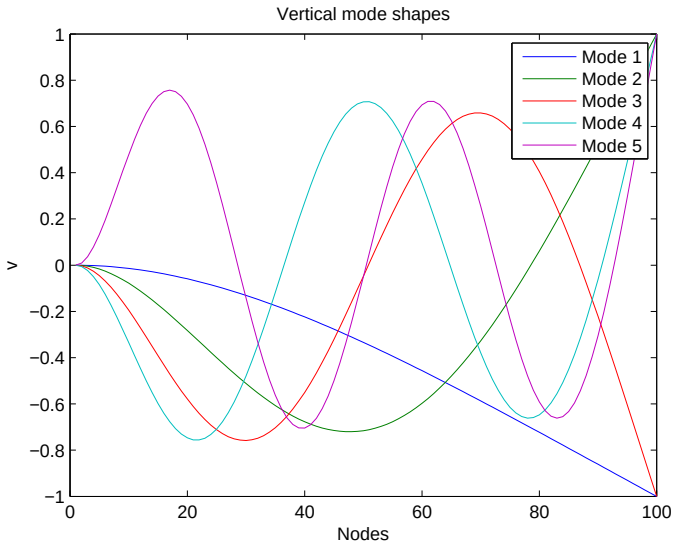

Figure 5.3. Mode shapes obtained from the FEM programmed in Matlab.

Moreover, these results are verified with Equation 5.2 used to calculate the frequencies of a cantilever beam.

$$f_i = \frac{{\alpha_i}^2}{2\pi}\sqrt{\frac{EI}{\rho A L^4}} \tag{5.2}$$

Where:

- i is the corresponding frequency.
- $\alpha_i = [1,875; \quad 4,694 \quad 7,853 \quad 10,996 \quad 14,137]$ for $i = 1, 2, 3, 4, 5$ respectively.

Frequency (Hz)	Equation 5.2	FEM
1	1,9773	1,9775
2	12,3926	12,3913
3	34,6856	34,6908
4	68,0060	67,9669
5	112,4067	112,3278

Table 5.5. Comparative table between the frequencies taken from Equation 5.2 and the ones from the FEM in order to verify the numerical model.

Table 5.5 shows the comparison between these two sets of frequencies and it can be appreciated that the frequencies obtained in the FEM are approximately the ones from the equation. Therefore, the Finite Element Model can be said to be correct, as the error comes from the number of elements.

5.1.3 Frequencies function

In order to be able to update the parameters of the FEM, this numerical model has been transformed into a function which calculates the natural frequencies of a beam with the values of the uncertain parameters given by the algorithms. It is basically the same script as in the FEM, but treated as a function and it needs the input of the number of frequencies to compare in the objective function and the chosen uncertain parameters of the numerical model.

5.2 Objective function

The objective function was introduced previously in Section 3.2 and it is one of the most important parts of Model Updating. An objective function is needed in order to be able to compare the behaviour of the numerical model and the experimental data and, therefore, the uncertain parameters must be changed so that the objective function is minimum.

Usually, the objective function involves the frequencies of the modes and the mode shapes. In the present Master Thesis, the objective function is related only to frequencies, as shown in Equation 5.3, in order to simplify the extraction of the

data from the experiments as well as it is the most reliable information that could be obtained. Nevertheless, it has to be mentioned that more accurate results should be obtained if the objective function contains information both about frequencies and mode shapes.

$$F = \sum_{i=1}^{m} \left(\frac{f_{exp}^{i} - f_{an}^{i}}{f_{exp}^{i}} \right)^2 \tag{5.3}$$

Where:

- m is the number of frequencies to compare.
- f_{exp}^{i} is the i^{th} experimental frequency.
- f_{an}^{i} is the i^{th} analytical frequency.

This function was obtained from Moradi et al. (2010), considering only the frequencies, and calculates the absolute error between the experimental and the analytical frequencies. The function is squared in order to avoid negative results that could lead to compensation of values when calculating the final sum in this objective function.

5.3 Bees Algorithm and Artificial Bee Colony algorithm

The first part of both algorithms consists of setting the upper and lower limits for the parameters to update.

Consequently, in order to calibrate the algorithms, the control parameters are also set, which are defined in the previous Sections 4.2.1 and 4.2.2. Different values of the control parameters were considered, choosing the ones which accomplished a good relation between producing precise results and being less time consuming. The options considered for the BA are extracted from Moradi et al. (2010) and Karaboga & Akay (2009a) and are summarized in Table 5.6.

Parameter	Moradi				Karaboga	
n	**10**	25	25	25	40	40
s	**5**	5	10	10	15	15
e	**1**	1	5	5	5	5
nsp	**3**	3	3	10	25	25
nep	**6**	6	6	15	25	25
ngh	**0,02**	0,02	0,02	0,02	0,5	3

Table 5.6. Control parameters analysed.

The tests were carried out with simulated data and considering only the Young's modulus as an uncertain parameter in order to simplify the problem. The results

showed that there was almost no improvement when increasing the values of the control parameters, although the time consumed was hugely superior. Therefore, the first combination of control parameters was finally implemented, which are in bold in Table 5.6.

Regarding the ABC algorithm, only one option was finally considered for the control parameters: $SN = 10$ and $limit = 3$ -which is set according to the maximum number of iterations-.

Furthermore, in order to be able to compare the results between both algorithms, the number of maximum iterations -or Maximum Cycle Number (MCN)- is chosen so that the number of evaluations of the frequency function is the same in both algorithms. Therefore, $MCN_{BA} = 18$ and $MCN_{ABC} = 15$.

Chapter 6

Experimental part

The experimental part consists of obtaining as many natural frequencies as possible of the analysed beam, in order to obtain more accurate results in the process of updating.

6.1 The beam and the tools

The structure used in the present Master Thesis is a "cantilever" steel beam. It has one free end and the other one is fixed with a C-clamp to a table and some weights, as shown in Figures 6.1 and 6.2, in order to fix it as best as possible.

The approximate characteristics of the beam are included in Table 5.1 from Section 5.1.2.

The process begins with the excitation of the beam with a hammer. The data acquisition is made using two accelerometers (Figure 6.3) at 2,5 cm -Accelerometer 1- and 60 cm -Accelerometer 2- from the free contour. The data is collected with the device *Spider8* and the program *Catman Professional*. The data gathered is then transformed into a MATLAB file using a *Catman file importer* function called *catman_read* by Dr. Andreas Geissler (Geissler, 2007). Finally, the accelerations are used in order to obtain the frequencies. All the tools needed for the experiment are shown on Figure 6.4.

Some characteristics had to be set to the *Catman Professional* program:

- Sample rate = 2400 Hz
- Variable filter = Bessel
- Frequency (low pass filter) = 300 Hz

These values were chosen according to the frequencies obtained in the programming of the FEM, so as to avoid aliasing.

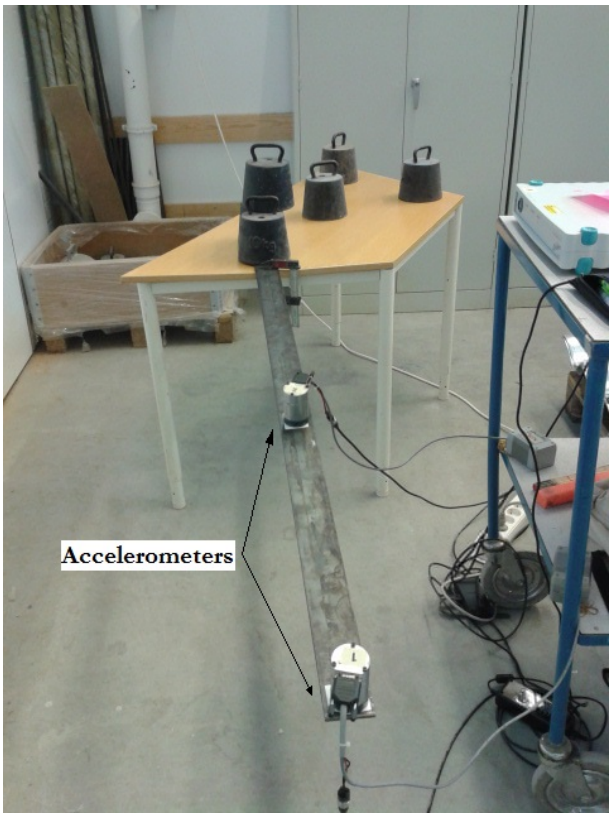

Figure 6.1. The beam used in the experimental part.

Figure 6.2. Detail of the fixed end with a C-clamp.

Figure 6.3. Detail of one accelerometer.

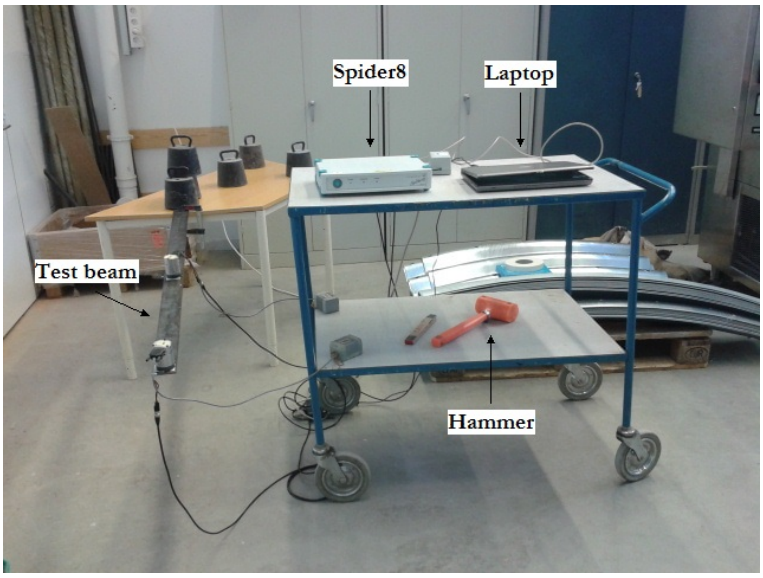

Figure 6.4. A picture of all the tools required for the experimental part.

6.2 Frequencies

Several trials were carried out in order to obtain the frequencies of the beam, in which an average has been done. At the initial stage, high frequencies could not be achieved. Nevertheless, the problem was solved hitting the beam less hard, more frequently and close to the fixed contour. The explanation resides in that:

- When hitting closer to the fixed node the higher frequencies are excited more than the lowest frequencies. On the other hand, hitting closer to the free node would basically excite the first node.
- When hitting more frequently, the signal is not allowed to be damped and a record of more length can be achieved. Therefore, the frequency spectrum is likely to be more accurate.
- Finally, when hitting less hard, the accelerometers are able to capture all the accelerations correctly, as it has an upper limit in which it cannot detect higher acceleration.

Moreover, if a third accelerometer would be placed near the fixed node, more accurate results from higher frequencies might be achieved, for the reasons mentioned above.

Examples of the results are shown on Figures 6.5 and 6.6. The first one shows the first frequency and the second one the higher ones. It is clear that only the first five frequencies were obtained accurately. Note that lower frequencies were best achieved with Accelerometer 1 and high frequencies with Accelerometer 2, because the displacement of the first mode longer at the free end.

Finally, the frequencies obtained are:

$$f = [2,06 \quad 12,00 \quad 38,70 \quad 73,30 \quad 124,60] \text{ Hz}$$

These frequencies are considered as correct, although there may be some errors due to the conditions in which the experiments were carried out. Therefore, the purpose of Model Updating in the present Master Thesis is to change the uncertain parameters in order that the frequencies obtained from the numerical model match as close as possible the experimental frequencies -by means of minimizing the objective function-.

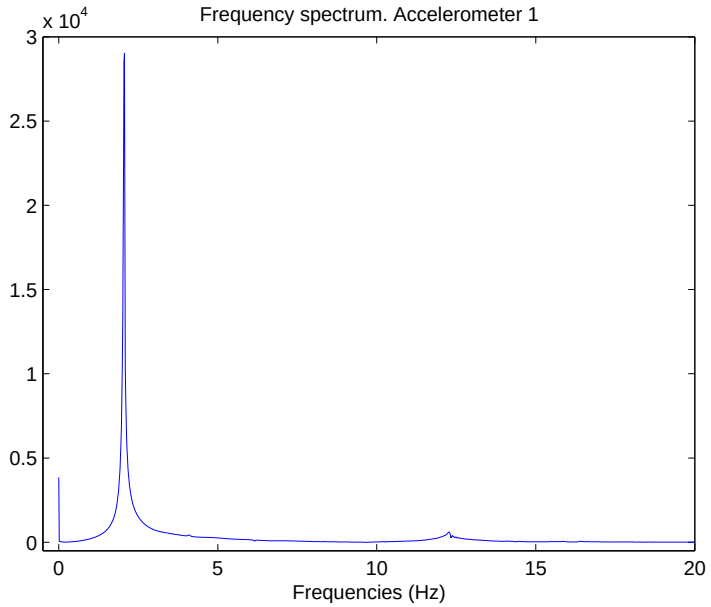

Figure 6.5. Frequency spectrum, first frequency. Accelerometer 1.

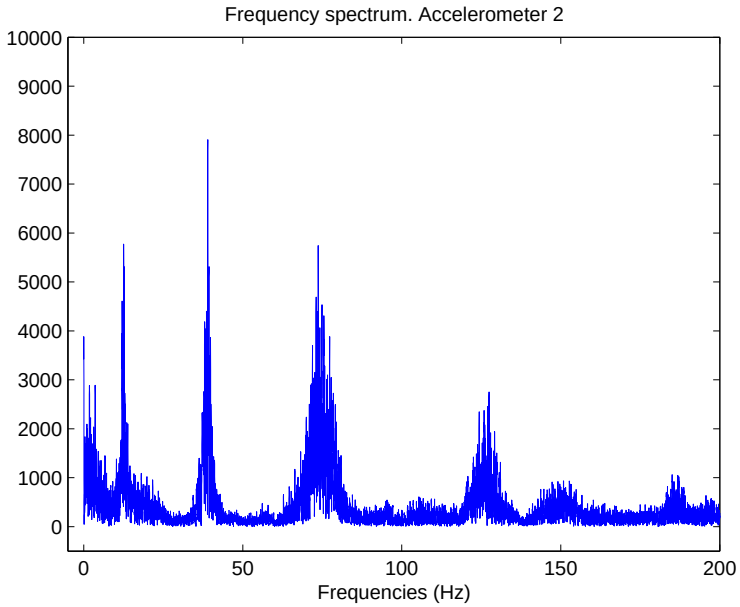

Figure 6.6. Frequency spectrum, higher frequencies. Accelerometer 2.

Chapter 7

The updating process

7.1 Frequencies comparison

Just before the updating process, a comparison of natural frequencies has to be performed, as explained in Section 2.3. These comparison can be made numerically (Table 7.1) or graphically (Figure 7.1). The comparison shows a clear tendency of the experimental frequencies to be higher than the analytical ones, with an exception of the second mode. Nevertheless, the similarity between the two models is perfectly clear, as it is a simple model.

Frequencies (Hz)	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
Experimental	2,06	12,00	38,70	73,30	124,60
Analytical	1,9775	12,3913	34,6908	67,9669	112,3278

Table 7.1. Comparative table between experimental and analytical frequencies.

7.2 Uncertain parameters

As first introduced in Chapter 2.1.2, one of the most important aspects in Model Updating is choosing the right uncertain parameters. They have to satisfy two requisites:

1. The frequencies must be sensitive to small changes in these parameters.
2. The parameters need to be the less reliable of all the parameters.

Moreover, their upper and lower boundaries have to be fixed, which must be coherent, as the algorithm cannot conclude with a solution without sense.

Given the conditions explained in the previous chapter, it seems obvious that the cantilever beam will not be perfect, as the supposed clamped end will not be completely fixed because the C-clamp cannot attach the beam perfectly to the

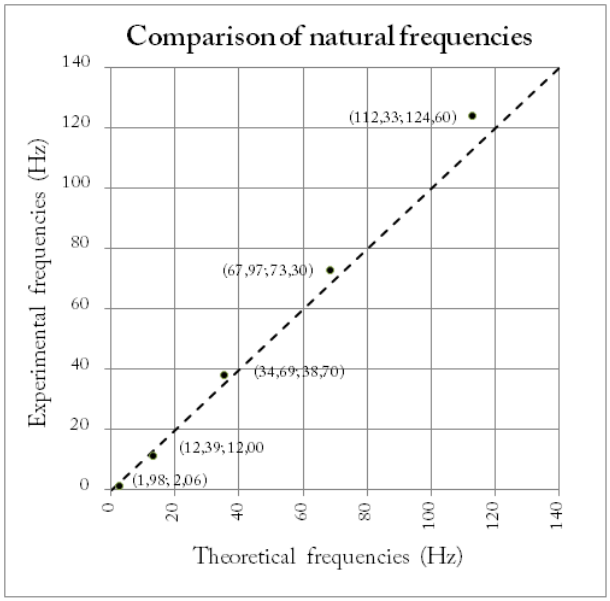

Figure 7.1. Comparative graph between experimental and analytical frequencies.

table. Thus, the initial objective of the programmed algorithms is to determine the horizontal, vertical and rotational stiffness of the beam in the fixed node, this is, to consider these three stiffnesses as the uncertain parameters.

The results obtained in this case are not satisfactory. It was observed that for high stiffnesses the effect on the frequencies is minimum. In other words, the algorithms could not distinguish between stiffnesses from 10^7 to infinite. As a consequence, the algorithms kept giving different results for each run with the same error and obtaining the same updated frequencies: the ones from a perfect cantilever beam. Nevertheless, the experimental frequencies were not similar to the ones from a perfect cantilever beam, as can be seen on Table 7.1.

Therefore, these stiffnesses could not be considered as uncertain parameters and other parameters should be chosen which were able to increase all the frequencies but lower the second one. Thus, their pattern of a "uniform cantilever beam" needed to be modified.

As the considered beam was slender, the effect of the mass of the sensors was then considered as a parameter that could highly influence the frequencies, lowering them. This effect could explain why the experimental second frequency was lower than the analytical one. Figure 7.2 shows the mode shape of the second frequency. The location of the two accelerometers was at nodes 53 and 97. In this case, the accelerometers could have a great influence on the second frequency, because the accelerometers are located where there is more displacement. Thus, they contribute giving more inertia to the system. This effect should also occur for the first frequency, although it is not appreciated in the experiments, or for the other higher

frequencies, but as the displacement is minor this effect is mitigated with the effect of other uncertain parameters that should also be taken into consideration.

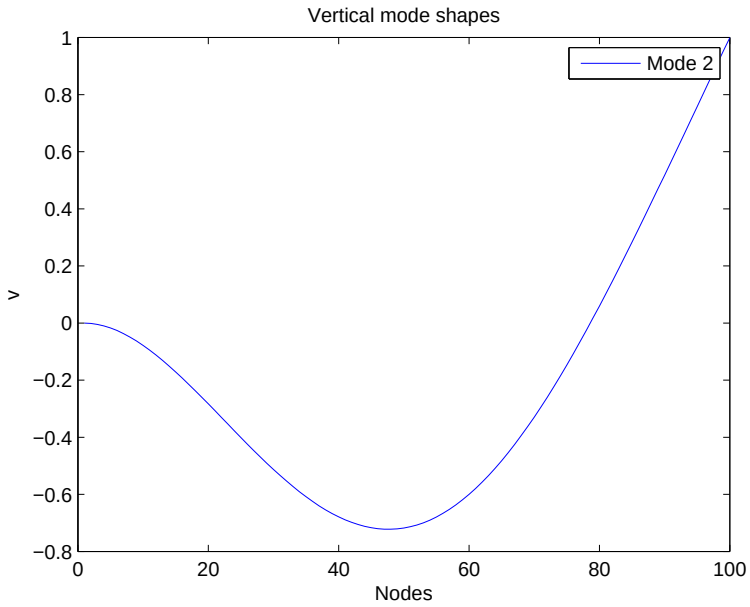

Figure 7.2. Second mode shape.

Therefore, the Finite Element Model was modified in order to include two point masses at nodes 53 and 97, Accelerometer 2 and 1 respectively.

Other parameters were analysed to be potential uncertain parameters and, finally, the parameters to update were the ones in Table 7.2.

Uncertain parameter	Notation	Range of values	Units	Frequency
Young's modulus	E	$1,9 \cdot 10^{11} - 2,1 \cdot 10^{11}$	N/m^2	↗
Steel density	ρ	$7750 - 8050$	kg/m^3	↘ ↘
Mass of each sensor	m	$0,15 - 0,25$	kg	↘
Thickness of the beam	T	$0,0035 - 0,0045$	m	↗ ↗

Table 7.2. Characteristics of the final uncertain parameters chosen.

As found in the literature, the Young's modulus of steel can vary between $1,9$ and $2,1 \cdot 10^{11} N/m^2$ and the steel density between 7750 and 8050 kg/m^3. The mass of the sensor was measured, $m = 0,16$ kg; nevertheless, this measurement did not include the plastic attached to the sensors, the adhesive tape used to attach the sensors to the beam or the cables connecting the sensors to the *Spider8* device. Finally, although it was measured, the thickness of the beam was also considered as an uncertain parameter due to the vast effect that it produced to the frequencies.

An increment of the thickness caused an increment to the moment of inertia, which at the same time had the effect of increasing enormously the frequencies.

As a consequence, the parameter *ngh* from the Bees Algorithm had to be reduced from 0,02 to 0,0002 units so that the neighbourhood search was within the boundaries. Thus, the *ngh* parameter was very small compared to the dimensions of the Young's modulus. This fact made the author consider transforming the *ngh* value into a vector which's values would depend on the dimension of each uncertain parameter -the wider the limits of the uncertain parameter, the higher the *ngh* could be-. Although it was dismissed as it was considered to be beyond the scope of the present Master Thesis, it could be a line for further research.

Again, it has to be stressed that the range of values in which the uncertain parameters can vary need to be plausible and coherent.

7.3 Results and analysis

An average of 10 runs was performed for each algorithm and the results obtained can be found on Table 7.3.

Algorithm		BA	ABC
Updated parameters	E (N/m^2)	$2,05 \cdot 10^{11}$	$2,06 \cdot 10^{11}$
	m (kg)	0,2372	0,2391
	ρ (kg/m^3)	7890,00	7894,20
	T (m)	0,0043	0,0043
Objective function		$1,3751 \cdot 10^{-2}$	$1,3751 \cdot 10^{-2}$
Updated frequencies (Hz)	f1 (2,06)	2,09	2,09
	f2 (12,00)	13,09	13,09
	f3 (38,70)	36,68	36,69
	f4 (73,30)	71,81	71,82
	f5 (124,60)	118,75	118,77

Table 7.3. Updated parameters.

Figures 7.3 and 7.4 show the results of each uncertain parameter, as well as the value of objective function, for each run. It can be seen that the results do not vary a lot, with exception of some isolated cases which could be solved by increasing the number of iterations of each run. Nevertheless, the value of the objective function is very similar in all runs and an average of the results is considered as good enough.

There is no indication of which algorithm performs better in the given conditions, as their results are very similar. Still, both seem to show quite good results, as the value of the objective function is substantially close to zero. Furthermore, the error of the Finite Element model with 99 elements is $2,76 \cdot 10^{-2}$ and, thus, intending to minimize even more the objective function would clearly not improve the results.

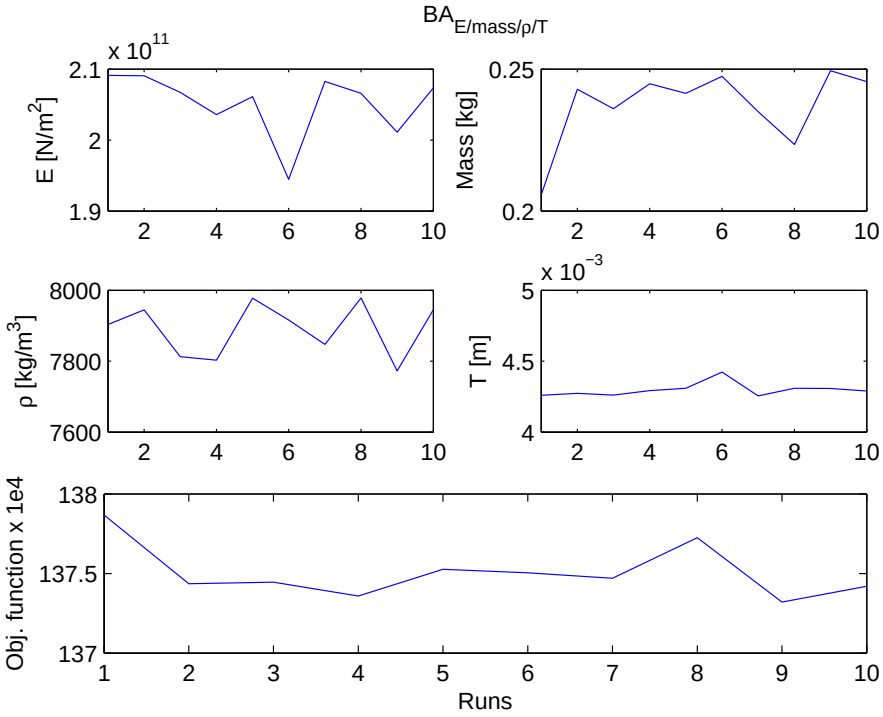

Figure 7.3. Results of each uncertain parameter and value of the objective function for each run using the BA.

Nevertheless, one point must be noted. Both algorithms showed a clear tendency to have the upper limit of the mass, 0,24 kg, even-though the measurements indicated that its weight was around 0,16 kg. As explained in the previous section, one reason could be the effect of all the additional mass from the plastic attached to the sensors, the adhesive tape used to attach the sensors to the beam or the cables connecting the sensors to the *Spider8* device, which were not weighted.

On the other hand, other tests were performed in order to look into how the algorithms would respond to higher upper limits for the mass. The results showed that the mass kept increasing as well as the thickness of the beam. The explanation resides in that these two parameters were trying to compensate each other in order to minimize the objective function. Again, the importance of reasonable limits is present.

Another issue that must be mentioned are the final frequencies. Figure 7.5 shows the graphical comparison between the experimental frequencies and the updated frequencies. Notice that the updated first frequency is slightly higher than the experimental one, the second frequency is approximately 1Hz higher and the rest are lower -the higher the mode, the bigger is the difference between updated and experimental frequencies-. This could be explained with the chosen objective function.

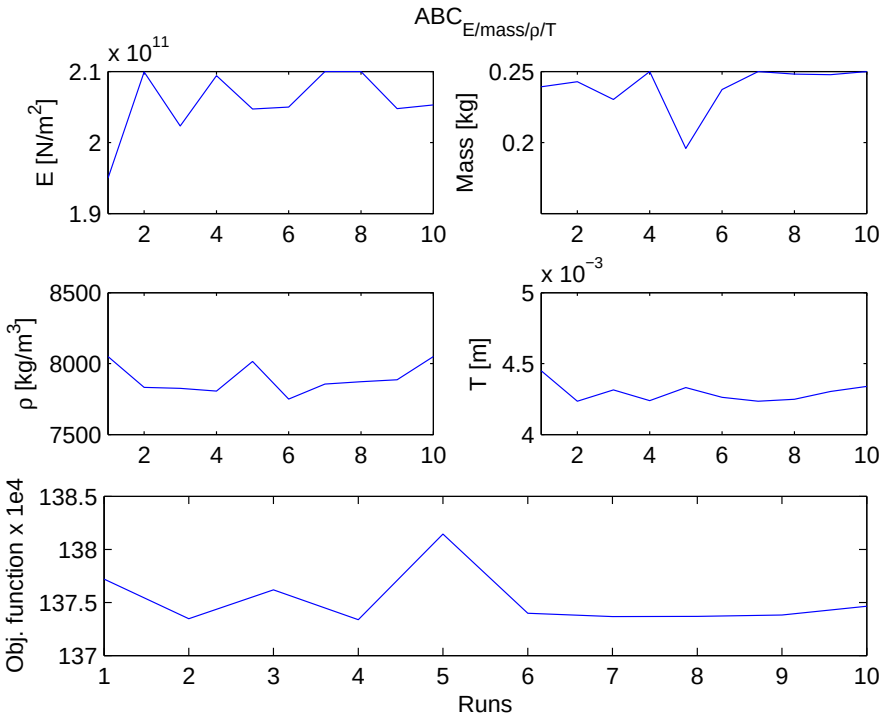

Figure 7.4. Results of each uncertain parameter and value of the objective function for each run using the ABC.

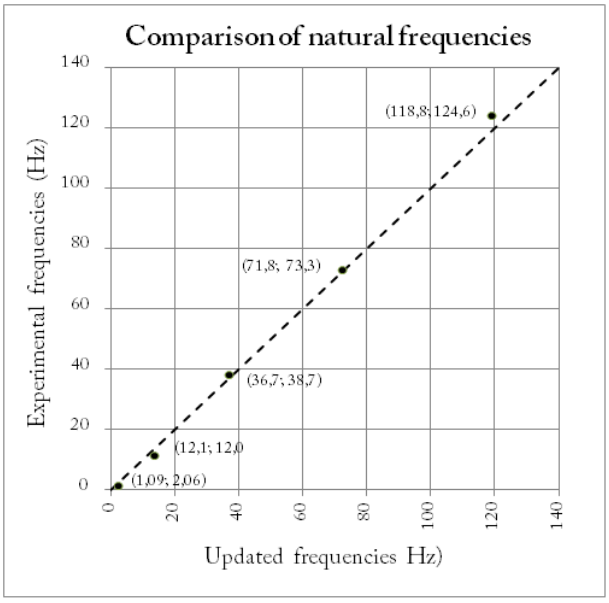

Figure 7.5. Comparative graph between experimental and updated frequencies.

The objective function is based on the relative error of the frequencies and, thus, the higher updated frequencies can have a larger difference compared to the experimental ones. Moreover, the algorithm is unable to distinguish between frequencies when updating, as its aim is to minimize the set of objective function and not its parameters separately. As a result, the algorithm tries to minimize the error as a whole.

The objective function might be improved including the mode shapes in the objective function. Nevertheless, the updated frequencies show a clear improvement from the initial ones.

Chapter 8

Conclusions and further work

8.1 Fulfilment of objectives

Once the project is developed, it is pertinent to verify the fulfilment of the posed objectives.

- A study of the theory of Model Updating is carried out.
- A first research of the most common algorithms in Model Updating is performed, deciding two new algorithms to implement.
- All basic processes in model updating were carried out, which include: programming the Finite Element Model, the two algorithms, arranging the real model, determining the real frequencies and finally updating the numerical model.
- The results are discussed and compared.

Therefore, the general objective of implementing and comparing two new algorithms in Finite Element Model Updating is fulfilled.

8.2 Conclusions

From Chapter 1:

- Model Updating fulfils the necessity of having a numerical model that represents the real behaviour and response of the structure.
- Model Updating has a vast importance when designing a new structure, as well as when controlling the performance of it once it is built, in order to optimize it and to avoid undesired situations that could lead to the collapse of the structure.

- It is important to continue studying new algorithms to implement and to improve the existing ones in order to enhance their performance and be more computationally efficient.

From Chapter 2:

- Measurements are usually imprecise and incomplete. Therefore, they are only a complement of the numerical model and the updated parameters do not need to exactly reproduce its behaviour.
- It is necessary to compare the numerical data with test results and obtain a good similarity between them in order to achieve good updated results.
- Many methods to compare the numerical data with tests results exist nowadays.

From Chapter 3:

- Despite having some disadvantages, overall, iterative methods seem to be better methods and are more used nowadays.
- It is difficult to classify all model updating methods, as they might combine different characteristics.
- Nature-inspired algorithms are emerging as new methods of optimizing functions and seem to have a good performance in model updating.
- The most used algorithms nowadays in model updating are Genetic Algorithms and Particle Swarm Optimization.
- Hybrid methods combine different model updating methods in order to improve its results.

From Chapter 4:

- Foraging bees have an organized and optimized way of searching food sources and gathering pollen, which has been used in order to develop two algorithms for optimizing multi-variable functions: Bees Algorithm and Artificial Bee Colony algorithm.
- Even these two algorithms are based on the same behaviour, they are completely different.

From Chapter 5:

- Many parameters need to be calibrated and validated to obtain numerical models and algorithms which work properly.

- The objective function is one of the most important parts in Model Updating, as it is the function that is to be minimized.
- The upper and lower boundaries of the uncertain parameters have a vast importance in order to obtain coherent results.
- In order to compare both algorithms objectively, the number of iterations of each algorithm has to be set so as their number of evaluations is the same.

From Chapter 6:

- Higher frequencies are more difficult to detect. In order to solve this problem, for the cantilever beam, the beam needs to be excited less hard, more frequently and closer to the fixed end.
- Lower frequencies -basically the first one- are better achieved by Accelerometer 1 and higher frequencies by Accelerometer 2. This is explained as the displacement that produces the first mode is larger at the free end and the contribution of the first node covers the other modes in the acquired signal.
- Preciser results might be achieved if a third accelerometer was placed close to the fixed end.

From Chapter 7:

- The comparison of frequencies shows a clear similarity between the two models, as the case of study is a simple beam.
- In order to choose the uncertain parameters, some tests need to be performed to determine the less reliable parameters and the ones which the frequencies are sensitive to.
- For higher vertical, horizontal and rotational stiffnesses of the fixed node, the frequencies are almost not sensitive to their changes. Thus, they cannot be uncertain parameters.
- The effect of the mass of the sensors needs to be considered, as the beam is slender and its weight is not too high compared to the one from the sensors. Its effect is lowering the frequencies, basically the second one, as the sensors are placed in a position where the second mode has its maximum displacement.
- Even-though the mass of the sensor was measured, it had to be considered as an uncertain parameter because cables, etc. could influence it.
- The thickness of the beam was considered as an uncertain parameter due to its high sensitivity on the frequencies, as there might be a small error when measuring it.

- Some parameters from the algorithms need to be changed when choosing the uncertain parameters in order to be coherent. This was the case of ngh from the BA when introducing the thickness of the beam as an uncertain parameter. It had to be diminished to be able to produce the neighbourhood search around the solution. A study might be carried out for BA with different ngh in each uncertain parameter according to its dimension -higher ngh for higher values of the uncertain parameters-.

- The updating process was carried out successfully, with a small value of the objective function. The error in the Finite Element Model due to the number of elements has to be accounted and, thus, a better precision is not necessary to be achieved.

- More studies should be carried out, but both algorithms could be successfully implemented in Model Updating for other structures.

- According to the results, it was not possible to decide which was the best algorithm between BA and ABC. Nevertheless, the BA was more difficult to calibrate due to its high number of setting parameters.

- The mass of the sensors is an example of how important is to set the upper and lower limits of the uncertain parameters, as erroneous values which compensate with each other could achieve a low value of the objective function although they are not coherent.

- The higher is the mode, the larger is the difference between updated and experimental frequencies. This is due to the chosen objective function, which is based on the relative error and, thus, the higher frequencies can have a larger difference in their value. Moreover, the chosen objective function cannot distinguish between individual frequencies, it minimizes the function as a whole.

8.3 Further work

As this study is very recent and given the scope of the present Master Thesis, still a lot of research needs to be done.

- Comparing these results with the results from currently used algorithms such as the Genetic Algorithm or the Particle Swarm Optimization.

- Implementing and testing the algorithms with the same beam, but with different boundary conditions. Finally, trying more complex structures.

- Producing damages to the cantilever beam and analysing the variation of the results. Will these algorithms be able to detect the location and magnitude of these damages?

- Considering to adapt the ngh from the Bees Algorithm as a vector, which's values would depend on the value of each uncertain parameter -the wider the limits of the uncertain parameter, the higher the ngh could be-.
- Trying more advanced features of these two algorithms, which can be found in the literature of their developers.

Bibliography

Aarts, E., & Korst, J. (1997). *Simulated annealing*. John Wiley & sons Ltd.

Allemang, R. J. (2003). The modal assurance criterion–twenty years of use and abuse. *Sound and Vibration*, *37*(8), 14–23.

Alvin, K. (1997). Finite element model update via bayesian estimation and minimization of dynamic residuals. *AIAA journal*, *35*(5), 879–886.

Baruch, M. (1982). Optimal correction of mass and stiffness matrices using measured modes. *AIAA Journal*, *20*(11), 1623–1626.

Baruch, M., & Itzhack, I. (1978). Optimal weighted orthogonalization of measured modes. *AIAA Journal*, *16*(4), 346–351.

Baykasoglu, A., Ozbakir, L., & Tapkan, P. (2007). *Artificial Bee Colony Algorithm and Its Application to Generalized Assignment Problem, Swarm Intelligence, Focus on Ant and Particle Swarm Optimization*. Itech Education and Publishing, Vienna, Austria.

Ben-Haim, Y., & Prells, U. (1993). Selective sensitivity in the frequency domain–i. theory. *Mechanical Systems and Signal Processing*, *7*(5), 461–475.

Berman, A. (1979). Comment on "optimal weighted orthogonalization of measured modes". *AIAA Journal*, *17*, 927–928.

Berman, A., & Nagy, E. (1983). Improvement of a large analytical model using test data. *AIAA Journal*, *21*(8), 1168–1173.

Börgesson, L. (1996). Abaqus. *Developments in geotechnical engineering*, *79*, 565–570.

Box, G., & Wilson, K. (1951). On the experimental attainment of optimum conditions. *Journal of the Royal Statistical Society. Series B (Methodological)*, *13*(1), 1–45.

Broyden, C. (1970). The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics*, *6*(1), 76–90.

Caesar, B. (1986). Update and identification of dynamic mathematical models. In *International Modal Analysis Conference, 4 th, Los Angeles, CA*, (pp. 394–401).

Caesar, B. (1987). Updating system matrices using modal test data. In *International Modal Analysis Conference, 5 th, London, England*, (pp. 453–459).

Chopra, A. (2001). *Dynamics of structures: Theory and applications to earthquake engineering*, vol. 2. Prentice Hall.

Ewins, D. (2000a). Adjustment or updating of models. *Sadhana*, *25*(3), 235–245.

Ewins, D. (2000b). *Modal testing: theory, practice and application*, vol. 2. Research studies press Baldock.

Ewins, D. (2000c). Model validation: Correlation for updating. *Sadhana*, *25*(3), 221–234.

Farhat, C., & Hemez, F. (1993). Updating finite element dynamic models using an element-by-element sensitivity methodology. *AIAA Journal*, *31*(9), 1702–1711.

Feng, F. Z., Kim, Y. H., & Yang, B.-S. (2006). Applications of hybrid optimization techniques for model updating of rotor shafts. *Structural and Multidisciplinary Optimization*, *32*(1), 65–75.

Fletcher, R. (1970). A new approach to variable metric algorithms. *The computer journal*, *13*(3), 317–322.

Fox, R., & Kapoor, M. (1968). Rates of change of eigenvalues and eigenvectors. *AIAA Journal*, *6*(12), 2426–2429.

Friswell, M., & Mottershead, J. (1995). *Finite element model updating in structural dynamics*, vol. 38. Springer.

Gavin, H. P. (2012). Structural element stiffness matrices and mass matrices. *CEE 541. Structural Dynamics. Duke University*.

Gazetas, G., Anastasopoulos, I., Gerolymos, N., Mylonakis, G., & Syngros, C. (2006). The collapse of the hanshin expressway (fukae) bridge, kobe 1995: Soil-foundation-structure interaction, reconstruction, seismic isolation. *Entwicklungen in der Bodenmechanik, Bodendynamik und Geotechnik*, *2*, 93–120.

Geissler, A. (2007). catman_read. In *Catman file importer*.
URL `http://www.mathworks.com/matlabcentral/fileexchange/6780`

Genovese, K., Lamberti, L., & Pappalettere, C. (2005). Improved global–local simulated annealing formulation for solving non-smooth engineering optimization problems. *International Journal of solids and Structures*, *42*(1), 203–237.

Gladwell, G. (2004). *Inverse problems in vibration*, vol. 119. Springer.

Goldfarb, D. (1970). A family of variable metric methods derived by variational means. *Mathematics of computation*, *24*(109), 23–26.

Green, D., & Unruh, W. G. (2006). The failure of the tacoma bridge: a physical model. *American journal of physics*, *74*, 706.

Holland, J. (1992). Genetic algorithms. *Scientific american*, *267*(1), 66–72.

Jafarkhani, R., & Masri, S. (2011). Finite element model updating using evolutionary strategy for damage detection. *Computer-Aided Civil and Infrastructure Engineering*, *26*(3), 207–224.

Jung, D., & Kim, C. (2011). Finite element model updating on small-scale bridge model using the hybrid genetic algorithm. *Structure and Infrastructure Engineering: Maintenance, Management, Life-Cycle Design and Performance*, *9*(5), 481–495.

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. Tech. rep., Techn. Rep. TR06, Erciyes Univ. Press, Erciyes.

Karaboga, D., & Akay, B. (2009a). Artificial bee colony (abc), harmony search and bees algorithms on numerical optimization. In *Proceedings of Innovative Produxction Machines and Systems Virtual Conference, IPROMS*.

Karaboga, D., & Akay, B. (2009b). A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation*, *214*(1), 108–132.

Karaboga, D., & Akay, B. (2009c). A survey: algorithms simulating bee swarm intelligence. *Artificial Intelligence Review*, *31*(1), 61–85.

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of Global Optimization*, *39*(3), 459–471.

Karaboga, D., & Basturk, B. (2008). On the performance of artificial bee colony (abc) algorithm. *Applied soft computing*, *8*(1), 687–697.

Kwon, Y. W., & Bang, H. (2000). *The finite element method using MATLAB*. CRC.

Levin, R., & Lieven, N. (1998). Dynamic finite element model updating using simulated annealing and genetic algorithms. *Mechanical Systems and Signal Processing*, *12*(1), 91–120.

Lieven, N. A., & Ewins, D. J. (1988). Spatial correlation of mode shapes, the coordinate modal assurance criterion (comac). In *Proceedings of the 6th International Modal Analysis Conference*, (pp. 690–695).

Lin, R., Lim, M., & Du, H. (1995). Improved inverse eigensensitivity method for structural analytical model updating. *Journal of vibration and acoustics*, *117*(2), 192–198.

Link, M., Weiland, M., & Barragan, J. (1987). Direct physical matrix identification as compared to phase resonance testing: An assessment based on practical application. In *International Modal Analysis Conference, 5 th, London, England, Proceedings.*, vol. 1, (pp. 804–811).

LUSAS (2010). *Application Examples Manual (Bridge, Civil & Structural).*

Martí, J. V., & González-Vidosa, F. (2010). Design of prestressed concrete precast pedestrian bridges by heuristic optimization. *Advances in Engineering Software*, *41*(7), 916–922.

Marwala, T. (2010). *Finite element model updating using computational intelligence techniques: applications to structural dynamics*. Springer.

Minas, C., & Inman, D. (1988). Correcting finite element models with measured modal results using eigenstructure assignment methods. In *International Modal Analysis Conference, 6 th, Kissimmee, FL*, (pp. 583–587).

Moaveni, S. (2003). *Finite element analysis: theory and application with ANSYS*. Pearson Education India.

Moradi, S., Fatahi, L., & Razi, P. (2010). Finite element model updating using bees algorithm. *Structural and Multidisciplinary Optimization*, *42*(2), 283–291.

Mottershead, J., & Friswell, M. (1993). Model updating in structural dynamics: a survey. *Journal of sound and vibration*, *167*(2), 347–375.

Mthembu, L., Marwala, T., Friswell, M., & Adhikari, S. (2011a). Finite element model selection using particle swarm optimization. *Dynamics of Civil Structures*, *4*, 41–52.

Mthembu, L., Marwala, T., Friswell, M. I., & Adhikari, S. (2011b). Model selection in finite element model updating using the bayesian evidence statistic. *Mechanical Systems and Signal Processing*, *25*(7), 2399–2412.

Pastor, M., Binda, M., & Harčarik, T. (2012). Modal assurance criterion. *Procedia Engineering*, *48*, 543–548.

Perera, R., & Ruiz, A. (2008). A multistage fe updating procedure for damage identification in large-scale structures based on multiobjective evolutionary optimization. *Mechanical Systems and Signal Processing*, *22*(4), 970–991.

Perez, R., & Behdinan, K. (2007). Particle swarm approach for structural design optimization. *Computers & Structures*, *85*(19), 1579–1588.

Pham, D., & Ghanbarzadeh, A. (2007). Multi-objective optimisation using the bees algorithm. In *Memorias del Innovative Production Machines and Systems Virtual Conference*.

Pham, D., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S., & Zaidi, M. (2005). The bees algorithm. technical note. Tech. rep., Manufacturing Engineering Centre, Cardiff University, UK.

Pham, D., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S., & Zaidi, M. (2006). The bees algorithm–a novel tool for complex optimisation problems. In *Proceedings of IPROMS 2006 conference*, (pp. 454–461).

Pham, D., Koc, E., Lee, J., & Phrueksanant, J. (2007). Using the bees algorithm to schedule jobs for a machine. In *Proceedings of eighth international conference on laser metrology, CMM and machine tool performance*, (pp. 430–439).

Rechenberg, I. (1965). Cybernetic solution path of an experimental problem. Tech. rep., Royal Air Force Establishment.

Ren, W., & Chen, H. (2010). Finite element model updating in structural dynamics by using the response surface method. *Engineering Structures*, *32*(8), 2455–2465.

Ribeiro, D., Calçada, R., Delgado, R., Brehm, M., & Zabel, V. (2012). Finite element model updating of a bowstring-arch railway bridge based on experimental modal parameters. *Engineering Structures*, *40*, 413–435.

Ross, R. (1971). Synthesis of stiffness and mass matrices from experimental vibration modes. In *SAE paper*, vol. 710787.

Salamon, P., Sibani, P., & Frost, R. (2002). *Facts, conjectures, and improvements for simulated annealing*, vol. 7. Society for Industrial & Applied Mathematics.

Schittkowski, K. (1986). Nlpql: A fortran subroutine solving constrained nonlinear programming problems. *Annals of operations research*, *5*(1), 485–500.

Schwefel, H. P. (1975). *Evolutionsstrategie und numerische optimierung*. Ph.D. thesis, Technical University Berlin.

Seeley, T. D. (1995). *The wisdom of the hive: the social physiology of honey bee colonies*. Harvard University Press.

Shanno, D. (1970). Conditioning of quasi-newton methods for function minimization. *Mathematics of computation*, *24*(111), 647–656.

Sonmez, F. O. (2007). Shape optimization of 2d structures using simulated annealing. *Computer methods in applied mechanics and engineering*, *196*(35), 3279–3299.

Strogatz, S. H., Abrams, D. M., McRobie, A., Eckhardt, B., & Ott, E. (2005). Theoretical mechanics: Crowd synchrony on the millennium bridge. *Nature*, *438*(7064), 43–44.

Thoren, A. (1972). Derivation of mass and stiffness matrices from dynamic test data. In *AIAA conference paper*, (pp. 72–346).

Tu, Z., & Lu, Y. (2008). Fe model updating using artificial boundary conditions with genetic algorithms. *Computers & Structures*, *86*(7), 714–727.

Wei (1989). Structural dynamic model modification using vibration test data. In *International Modal Analysis Conference, 7 th, Las Vegas, Nevada*, (pp. 562–567).

Yang, X.-S. (2005). Engineering optimizations via nature-inspired virtual bee algorithms. *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach*, *3562*, 317–323.

Yuen, K.-V. (2010). *Bayesian methods for structural dynamics and civil engineering*. Wiley.

Zang, H., Zhang, S., & Hapeshi, K. (2010). A review of nature-inspired algorithms. *Journal of Bionic Engineering*, *7*, S232–S237.

Zhang, E., Feissel, P., & Antoni, J. (2011). A comprehensive bayesian approach for model updating and quantification of modeling errors. *Probabilistic Engineering Mechanics*, *26*(4), 550–560.

Zheng, Y.-m., Sun, H.-h., Zhao, X., Chen, W., Zhang, R.-h., & Shen, X.-d. (2009). Finite element model updating of a long-span steel skybridge. *Journal of Vibration Engineering*, *1*, 019.

Appendix A

MATLAB files

A.1 Finite Element Model

```
%  Roser Marrè Badalló  %

% This file is the initial Finite Element Model of the beam.

clear all

%% Data

m = 5; %Number of frequencies to compare [-]
L = 1.3; %Length of the beam [m]
Ne = 99; %Number of elements [-]
No = Ne+1; %Number of nodes [-]
ndof = (Ne+1)*3; %Number of degrees of freedom
Le = L/Ne; %Length of the elements [m]
W = 0.06; %Width of the beam [m]
T = 0.004; %Thickness of the beam [m]
e1 = 53;
e2 = 97;

E = 2.10e11; %Modulus of elasticity [N/m^2]
rho = 7.850e3; %Density of the steel [kg/m^3]
A = W*T; %Area of the section of the beam [m^2]
I = 1/12*W*T^3; %Moment of inertia [m^4]
mass = 0.16; %Mass of each sensor [kg]

kh1 = 1e10; %stiffness of the horizontal spring in node one
kh2 = 0; %stiffness of the horizontal spring in node ndof
kv1 = 1e10; %stiff vertical node 1
kv2 = 0; %stiff vertical node ndof
kO1 = 1e10; %stiff rotation node 1
kO2 = 0; %stiff rotation node ndof
beta = 0.5; %"average" between lumped and consistent element mass matrix
```

```
%% Stiffness and mass matrices:

% initialization of stiffnes and mass matrix:
K = zeros(ndof);
M = zeros(ndof);

% element stiffness matrix:
Ke = [A*E/Le 0            0              A*E/Le 0             0
      0      12*E*I/Le^3  -6*E*I/Le^2    0      -12*E*I/Le^3  -6*E*I/Le^2
      0      -6*E*I/Le^2  4*E*I/Le       0      6*E*I/Le^2    2*E*I/Le
      A*E/Le 0            0              A*E/Le 0             0
      0      -12*E*I/Le^3 6*Le*E*I/Le^3 0      12*E*I/Le^3   6*E*I/Le^2
      0      -6*E*I/Le^2  2*E*I/Le       0      6*E*I/Le^2    4*E*I/Le];

% element consistent mass matrix:
Mec = rho*A*Le/420 * ...
    [140    0       0          70      0       0
     0      156     -22*Le     0       54      13*Le
     0      -22*Le  4*Le^2     0       -13*Le  -3*Le^2
     140    0       0          70      0       0
     0      54      -13*Le     0       156     22*Le
     0      13*Le   -3*Le^2    0       22*Le   4*Le^2];

% element lumped mass matrix:
Mel = rho*A*Le/2*...
    [1  0   0       0   0   0
     0  1   0       0   0   0
     0  0   Le^2/12  0   0   0
     0  0   0       1   0   0
     0  0   0       0   1   0
     0  0   0       0   0   Le^2/12];

% element "average" mass matrix:
Me = (1-beta)*Mel+beta*Mec;

% element mass matrix of the sensors:
Mmass = mass*Le/420 * ...
    [140    0       0          70      0       0
     0      156     -22*Le     0       54      13*Le
     0      -22*Le  4*Le^2     0       -13*Le  -3*Le^2
     140    0       0          70      0       0
     0      54      -13*Le     0       156     22*Le
     0      13*Le   -3*Le^2    0       22*Le   4*Le^2];

% assembly of matrices:
for j = 1:Ne
    % degrees of freedom for element j
    dofs = (j-1)*3+1:(j+1)*3;
    % add stiffnes matrix of element j
    K(dofs,dofs) = K(dofs,dofs) + Ke;
    % add mass matrix of element j
    M(dofs,dofs) = M(dofs,dofs) + Me;
end
```

```
 for j = e1
     dofs = (j-1)*3+1:(j+1)*3;
     M(dofs,dofs) = M(dofs,dofs) + Mmass;
 end
 for j = e2
     dofs = (j-1)*3+1:(j+1)*3;
     M(dofs,dofs) = M(dofs,dofs) + Mmass;
 end

%% Springs (boundary conditions):
K(1,1) = K(1,1) + kh1;
K(2,2) = K(2,2) + kv1;
K(3,3) = K(3,3) + kO1;
K(ndof-2,ndof-2) = K(ndof-2,ndof-2) + kh2;
K(ndof-1,ndof-1) = K(ndof-1,ndof-1) + kv2;
K(ndof,ndof) = K(ndof,ndof) + kO2;

%% Equation of motion:
%[K-W^2M]u=0 (undamped system)
%phi: mode shapes
%W2: W^2 (square frequencies in a diagonal matrix)

[phi,W2]=eig(K,M);

w2 = diag(W2);
w=sqrt(w2);
[w,index] = sort(w); % Sorts my magnitude the frequencies (- to +)
phi = phi(:,index); % Sort the mode shapes according to the frequencies
f=w/(2*pi);

% Creating a vector with the mth first frequencies:
fm = zeros(m,1);
for i=1:m;
    fm(i,1) = f(i,1);
end

disp(fm)

%% Mode shapes

% Horitzontal:
phiu=zeros(No,No);
for i=1:No;
    for j=1:No;
        phiu(i,j)=phi(i*3-2,j);
    end
end

% Vertical:
phiv=zeros(No,No);
for i=1:No;
    for j=1:No;
        phiv(i,j)=phi(i*3-1,j);
    end
```

```
end

% Rotation:
phio=zeros(No,No);
for i=1:No;
    for j=1:No;
        phio(i,j)=phi(i*3,j);
    end
end

%% Plots

% Normalization of vertical modes:
phivN = zeros(No,No);

for j=1:No
    phiabs = abs(phiv(:,j));
    phimax = max(phiabs);
    phivN(:,j)=phiv(:,j)/phimax;
end

% Taking only the first mth vertical modes:
phiv10=zeros(No,m);
for i=1:No;
    for j=1:m;
        phiv10(i,j)=phivN(i,j);
    end
end

% Normalization of vertical modes:
phiON = zeros(No,No);

for j=1:No
    phiabs = abs(phio(:,j));
    phimax = max(phiabs);
    phiON(:,j)=phio(:,j)/phimax;
end

% Taking only the first mth vertical modes:
phiO10=zeros(No,m);
for i=1:No;
    for j=1:m;
        phiO10(i,j)=phiON(i,j);
    end
end

% Ploting vertical mode shapes:
fig1 = figure(1);
plot(phiv10)
title('Vertical mode shapes')
ylabel('v')
xlabel('Nodes')
legend('Mode 1','Mode 2','Mode 3','Mode 4','Mode 5')
```

A.2 Frequencies function

```
%  Roser Marrè Badalló  %

% This function calculates the natural frequencies of the beam.
% Inputs (uncertain parameters to update):
%   m = number of calculated frequencies
%   E = Young's modulus [N/m^2]
%   mass = mass of the sensors [kg]
%   rho = density of the beam [kg/m^3]
%   T = thickness of the beam [m]

function [f10] = frequenciesEmrT(m, E, mass, rho, T)

%% Data

L = 1.3; %length of the beam [m]
Ne = 99; %number of elements [-]
No = Ne+1; %number of nodes [-]
ndof = (Ne+1)*3; %number of degrees of freedom [-]
Le = L/Ne; %length of the elements [m]
W = 0.06; %width of the beam [m]
%T = 0.004; %thickness of the beam [m]
e1 =54;
e2 = 97;

%E = 2.1e11; %modulus of elasticity [N/m^2]
A = W*T; %area of the section of the beam [m^2]
I = 1/12*W*T^3; %moment of inertia [m^4]
%rho = 7850; %density of the steel [kg/m^3]
%mass = 0.150;

kh1 = 2e7; %horizontal stiffness
kh2 = 0;
kv1 = 2e7; %vertical stiffness
kv2 = 0;
kO1 = 2e7; %rotational stiffness
kO2 = 0;
beta = 0.5; %"average" between lumped and consistent element mass matrix

%% Stiffnes and mass matrices:

% initialization of stiffnes and mass matix:
K = zeros(ndof);
M = zeros(ndof);

% element stiffness matrix:
Ke = [A*E/Le 0            0              A*E/Le  0            0
      0      12*E*I/Le^3  -6*E*I/Le^2    0       -12*E*I/Le^3 -6*E*I/Le^2
      0      -6*E*I/Le^2  4*E*I/Le       0       6*E*I/Le^2   2*E*I/Le
      A*E/Le 0            0              A*E/Le  0            0
      0      -12*E*I/Le^3 6*Le*E*I/Le^3  0       12*E*I/Le^3  6*E*I/Le^2
```

```
      0      -6*E*I/Le^2  2*E*I/Le      0       6*E*I/Le^2   4*E*I/Le];

% element consistent mass matrix:
Mec = rho*A*Le/420 * ...
    [140    0       0          70      0       0
     0      156     -22*Le     0       54      13*Le
     0      -22*Le  4*Le^2     0       -13*Le  -3*Le^2
     140    0       0          70      0       0
     0      54      -13*Le     0       156     22*Le
     0      13*Le   -3*Le^2     0       22*Le   4*Le^2];

% element lumped mass matrix:
Mel = rho*A*Le/2*...
    [1  0   0       0   0   0
     0  1   0       0   0   0
     0  0   Le^2/12  0   0   0
     0  0   0       1   0   0
     0  0   0       0   1   0
     0  0   0       0   0   Le^2/12];

% element "average" mass matrix:
Me = (1-beta)*Mel+beta*Mec;

% element mass matrix of the sensors:
Mmass = mass*Le/420 * ...
    [140    0       0          70      0       0
     0      156     -22*Le     0       54      13*Le
     0      -22*Le  4*Le^2     0       -13*Le  -3*Le^2
     140    0       0          70      0       0
     0      54      -13*Le     0       156     22*Le
     0      13*Le   -3*Le^2     0       22*Le   4*Le^2];

% assembly of matrices:
for j = 1:Ne
    % degrees of freedom for element j
    dofs = (j-1)*3+1:(j+1)*3;
    % add stiffnes matrix of element j
    K(dofs,dofs) = K(dofs,dofs) + Ke;
    % add mass matrix of element j
    M(dofs,dofs) = M(dofs,dofs) + Me;
end

 for j = e1
     dofs = (j-1)*3+1:(j+1)*3;
     M(dofs,dofs) = M(dofs,dofs) + Mmass;
 end
 for j = e2
     dofs = (j-1)*3+1:(j+1)*3;
     M(dofs,dofs) = M(dofs,dofs) + Mmass;
 end

%% "Springs" (boundary conditions):
K(1,1) = K(1,1) + kh1;
K(2,2) = K(2,2) + kv1;
```

```
K(3,3) = K(3,3) + kO1;
K(ndof-2,ndof-2) = K(ndof-2,ndof-2) + kh2;
K(ndof-1,ndof-1) = K(ndof-1,ndof-1) + kv2;
K(ndof,ndof) = K(ndof,ndof) + kO2;

%% Equation of motion:
%[K-W^2M]u=0 (undamped system)

%phi: mode shapes
%W2: W^2 (square frequencies in a diagonal matrix)

[phi,W2]=eig(K,M);

w2 = diag(W2);
w = sqrt(w2);
[w,index] = sort(w); %sorts by magnitude the frequencies (- to +)
phi = phi(:,index); %sort the mode shapes according to the frequencies
f = w/(2*pi);

% Taking only the first 6 frequencies
f10 = zeros(m,1);
for i=1:m;
    f10(i,1)=f(i);
end

end
```

A.3 Bees Algorithm

```
% Roser Marrè Badalló %

% This file is the Bees Algorithm.

close all
tic
%% Design parameters (boundaries):

dp = 4; %number of design parameters

Emin = 1.9e11;
Emax = 2.1e11;

massmin = 0;
massmax = 1;

rhomin = 7750;
rhomax = 8050;

Tmin = 0.0035;
Tmax = 0.0045;

%% Objective function:

% F = sum (i=1,m)((fm-fa)/fm)^2
m = 5; % number of modes to compare in the objective function
% fm: experimental frequency (real/measured)
fm =[2.06; 12.00; 38.70; 73.30; 124.60];
% fa: analytical frequency (to update)

%% Control parameters:

N = 10; % Number of total solutions (initially random)
N1 = 5; % Number of best solutions
N2 = 1; % Number of elite solutions
n1 = 3; % Number of neighbour searches around each best solution
n2 = 6; % Number of neighbour searches around each elite solution
r = 0.0002; % Radius of the initial neighbour search
R = N-N1; % Number of random solutions

%% 1. Initialize a random population of N solutions:

MinE = ones(1,N)*Emin;
MaxE = ones(1,N)*Emax;
Minmass = ones(1,N)*massmin;
Maxmass = ones(1,N)*massmax;
Minrho = ones(1,N)*rhomax;
Maxrho = ones(1,N)*rhomin;
MinT = ones(1,N)*Tmin;
MaxT = ones(1,N)*Tmax;
```

```
Min = [MinE; Minmass; Minrho; MinT];
Max = [MaxE; Maxmass; Maxrho; MaxT];

% Random solutions:
% a = rand(0,1);
% xrand = xmin+a*(xmax-xmin);

Ran = Min+rand(dp,N).*(Max-Min);

%% 2. Evaluate the fitness of the population:

% 2a) Calculate the frequencies using the model function.
f = zeros(m,N);
for p=1:N
[f(:,p)]=frequenciesEmrT(m, Ran(1,p), Ran(2,p), Ran(3,p), Ran(4,p));
end

% 2b) Calculate the values of the objective function.
F=zeros(1,N);
for i=1:m
    F = F+((f(i,:)-fm(i)*ones(1,N))/fm(i)).^2;
end

%% 3. Select best solutions
MCN = 18;
it = 0;
%FF = 100*ones(1,MCN);
%RR = zeros(3,MCN);

% 3a) Sort solutions by the error:
[Fs,index] = sort(F); % Sorts by magnitude the errors (- to +)
Rans = Ran(:,index); % Sorts the kv1 & kv2 according to the errors.

while it<MCN
    % 3b) Select the best N1 solutions:
    RN1 = zeros(dp,N1);
    for i=1:dp
        for j=1:N1
            RN1(i,j) = Rans(i,j);
        end
    end

    % 3c) Select N2 elite solutions out of the N1 best solutions
    RN2 = zeros(dp,N2);
    for i=1:dp
        for j=1:N2
            RN2(i,j)=Rans(i,j);
        end
    end

    %% 4. Neighbourhood searching around each element of the solution
    % vector and selecting the best solution in each neighbourhood space:
    % xp(i)=(x(i)-r)+2*a(i)*r
```

```
    % 4a) Around the best solutions:
    % I have N1 solutions and for each one I have to do a n1 neighbour
    % search.

    Nbest1 = zeros(dp,N1);
    for q=1:N1 %q represents each N1 solution which will have a
        % neighbourhood
        Ne1 = zeros(dp,n1); %generate n1 neighbours from each N1(q)
        for j=1:n1
            Ne1(:,j)=(RN1(:,q)-r.*ones(dp,1))+2*rand(dp,1).*r;
        end

        %3b) if v is out of boundaries, it is shifted onto the boundaries
        for i=1
            for j = 1:n1
                if Ne1(i,j)<Emin
                   Ne1(i,j) = Emin;
                elseif Ne1(i,j)>Emax
                    Ne1(i,j) = Emax;
                end
            end
        end
        for i=2
            for j = 1:n1
                if Ne1(i,j)<massmin
                   Ne1(i,j) = massmin;
                elseif Ne1(i,j)>massmax
                    Ne1(i,j) = massmax;
                end
            end
        end
        for i=3
            for j = 1:n1
                if Ne1(i,j)<rhomin
                    Ne1(i,j) = rhomin;
                elseif Ne1(i,j)>rhomax
                    Ne1(i,j) = rhomax;
                end
            end
        end
        for i=4
            for j = 1:n1
                if Ne1(i,j)<=Tmin
                    Ne1(i,j) = Tmin;
                elseif Ne1(i,j)>=Tmax
                    Ne1(i,j) = Tmax;
                end
            end
        end

        f1 = zeros(m,n1); %calculate their frequencies
        for p=1:n1
[f1(:,p)]=frequenciesEmrT(m, Ne1(1,p), Ne1(2,p), Ne1(3,p), Ne1(4,p));
```

```
        end

        F1 = zeros(1,n1); %calculate the error
        for i=1:m
            F1 = F1 + ((f1(i,:)-fm(i)*ones(1,n1))/fm(i)).^2;
        end

        [Fs1,index] = sort(F1); % Sorts by magnitude the errors (- to +)
        Ne1b = Ne1(:,index); % Sorts the design parameters according to
        % the errors.
        N1best = Ne1b(:,1); %we want the best of the neighbourhood q
        if Fs1(1)>Fs(q) %to make sure that the new neighbour solution is
            % better than the "original one" that began the search
            Fs1(1)=Fs(q);
            N1best = Rans(:,q);
        end
        Nbest1(:,q)=N1best(:,1); %includes all the best solutions of each
        % neighbourhood search space
        Fs1b(q)=Fs1(1); % corresponding errors
    end

    % 4b) Around the elite solutions:
    % I have N2 solutions and for each one I have to do a n2 neighbour
    % search.

    Nbest2 = zeros(dp,N2);
    for q=1:N2 %q represents each N2 solution which will have a
        % neighbourhood
        Ne2 = zeros(dp,n2); %generate n2 neighbours from each N2(q)
        for j=1:n2
            Ne2(:,j)=(RN2(:,q)-r.*ones(dp,1))+2*rand(dp,1).*r;
        end

        %3b) if v is out of boundaries, it is shifted onto the boundaries
         for i=1
            for j = 1:n1
                if Ne1(i,j)<Emin
                   Ne1(i,j) = Emin;
                elseif Ne1(i,j)>Emax
                    Ne1(i,j) = Emax;
                end
            end
        end
        for i=2
            for j = 1:n2
                if Ne2(i,j)<massmin
                    Ne2(i,j) = massmin;
                elseif Ne2(i,j)>massmax
                    Ne2(i,j) = massmax;
                end
            end
        end
       for i=3
            for j = 1:n1
```

```
                if Ne1(i,j)<rhomin
                    Ne1(i,j) = rhomin;
                elseif Ne1(i,j)>rhomax
                    Ne1(i,j) = rhomax;
                end
            end
        end
        for i=4
            for j = 1:n1
                if Ne2(i,j)<=Tmin
                    Ne2(i,j) = Tmin;
                elseif Ne2(i,j)>=Tmax
                    Ne2(i,j) = Tmax;
                end
            end
        end

        f2 = zeros(m,n2); %calculate their frequencies
        for p=1:n2
[f2(:,p)]=frequenciesEmrT(m, Ne2(1,p), Ne2(2,p), Ne2(3,p), Ne2(4,p));
        end

        F2 = zeros(1,n2); %calculate the error
        for i=1:m
            F2 = F2 + ((f2(i,:)-fm(i)*ones(1,n2))/fm(i)).^2;
        end

        [Fs2,index] = sort(F2); % Sorts by magnitude the errors (- to +)
        Ne2b = Ne2(:,index); % Sorts parameters according to the errors
        N2best = Ne2b(:,1); %we want the best of the neighbourhood q
        if Fs2(1)>Fs(q) %to make sure that the new neighbour solution is
            % better than the "original one" that began the search
            Fs2(1)=Fs(q);
            N2best = Rans(:,q);
            %r = r*0.8;
        end
        Nbest2(:,q)=N2best(:,1); %includes all the best solutions of each
        % neighbourhood search space
        Fs2b(q)=Fs2(1); %corresponding errors
    end

    % We need to know which is best: Nbest1(:,q) or Nbest2(:,q),
    % which are from the same search space!
    for q=1:N2
        if Fs2b(q)<Fs1b(q)
            Fs1b(1,q) = Fs2b(1,q);
            Nbest1(:,q) = Nbest2(:,q); %In this way, we have the best
            % solutions of each search space included in Nbest1
            % (obtaining N1 new solutions).
        end
    end

    %% 5. Select the remaining R=N-N1 solutions randomly and evaluate
    % their fitnesses:
```

```
MinER = ones(1,R)*Emin;
MaxER = ones(1,R)*Emax;

MinmassR = ones(1,R)*massmin;
MaxmassR = ones(1,R)*massmax;

MinrhoR = ones(1,R)*rhomin;
MaxrhoR = ones(1,R)*rhomax;

MinTR = ones(1,R)*Tmin;
MaxTR = ones(1,R)*Tmax;

MinR = [MinER; MinmassR; MinrhoR; MinTR];
MaxR = [MaxER; MaxmassR; MaxrhoR; MaxTR];

Ra = MinR+rand(dp,R).*(MaxR-MinR);

fr = zeros(m,R); % Calculate the frequencies using the model function
for p=1:R
[fr(:,p)]=frequenciesEmrT(m, Ra(1,p), Ra(2,p), Ra(3,p), Ra(4,p));
end

Fr = zeros(1,R); % Calculate the error
for i=1:m
    Fr = Fr + ((fr(i,:)-fm(i)*ones(1,R))/fm(i)).^2;
end

%% 6. Form the new population using the solutions obtained
% (Nbest1, Nbest2 and Ra):
Ran = [Nbest1 Ra]; %Nbest2 included in Nbest1
F =[Fs1b Fr]; %These are all the errors, so we can sort them.
% Observation: We have put the name of Ran & F of the begining
% so that we can do the "while" again

it = it+1
%r = r*0.8;

%% 7. Choose the best solution:
[Fs,index] = sort(F); % Sorts by magnitude the errors (- to +)
Rans = Ran(:,index); % Sorts the kv according to the errors.

%% 8. Reduce the r if there is no improvement in the last 3
% iterations
% (This is a more advanced Bees Algoritm, not used in the Master
% Thesis)
%FF(1,it) = Fs(1); %Store the error of each iteration in a matrix
%RR (:,it) = Rans(:,1);% Store the results obtained at each iteration

%if it>3
%    if FF(1,it) >= FF(1,it-1) && FF(1,it) >= FF(1,it-2)
%       && FF(1,it) >= FF(1,it-3)
%        r = r*0.9;
%    end
```

```
    %end

    Fs(1);
    Rans(:,1);

end

xbest = Rans(:,1);
[f]=frequenciesEmrT(m, Rans(1,1), Rans(2,1), Rans(3,1), Rans(4,1));
fbest = f;
Fbest = Fs(1);

toc
```

A.4 Artificial Bee Colony algorithm

```
% Roser Marrè Badalló %

% This file is the Artificial Bee Colony Algorithm.

close all
tic

%% Design paramaters:

D = 4; %number of design parameters (dimension of the problem)

Emin = 1.9e11;
Emax = 2.1e11;

massmin = 0.15;
massmax = 0.25;

rhomin = 7750;
rhomax = 8050;

Tmin = 0.0035;
Tmax = 0.0045;

min = [Emin; massmin; rhomin; Tmin];
max = [Emax; massmax; rhomax; Tmax];

%% Objective function:

% F = sum (i=1,m)((fm-fa)/fm)^2
m = 5; % number of modes to compare in the objective function
% fm: experimental frequency (real/measured)
fm =[2.06; 12.00; 38.70; 73.30; 124.60];
% fa: analytical frequency (to update)

%% Control parameters:

TN = 20;
SN = TN/2; %total number of solutions (food sources)
% =n° of employed/onlooker bees
MCN = 15; %maximum iterations (maximum cycle number)
limit = 3; %limit control parameter

%% 1. Initialize a random population of SN solutions
%(in the range of parameters):

MinE = ones(1,SN)*Emin;
MaxE = ones(1,SN)*Emax;
Minmass = ones(1,SN)*massmin;
Maxmass = ones(1,SN)*massmax;
Minrho = ones(1,SN)*rhomin;
```

```
Maxrho = ones(1,SN)*rhomax;
MinT = ones(1,SN)*Tmin;
MaxT = ones(1,SN)*Tmax;

Min = [MinE; Minmass; Minrho; MinT];
Max = [MaxE; Maxmass; Maxrho; MaxT];

x = Min+rand(D,SN).*(Max-Min);

%% 2. Evaluate the fitness of the population:

%2a) Calculate the frequencies using the model function
f0 = zeros(m,SN);
for p=1:SN
    [f0(:,p)]=frequenciesEmrT(m, x(1,p), x(2,p), x(3,p), x(4,p));
end

%2c) Calculate the values of the objective function.
F0=zeros(1,SN);
for i=1:m
    F0 = F0+((f0(i,:)-fm(i)*ones(1,SN))/fm(i)).^2;
end

%% 3. Employed bees:

it = 0;
cycle = zeros(1,SN);
FF = zeros(1,MCN*2); %in order to save all the best solutions of each
% iteration
xx = zeros(D,MCN*2);
ff = zeros(m,MCN*2);

while it<MCN
    %3a) produce new solutions v for the employed bees and evaluate them
    for i = 1:D
        for j = 1:SN
            k = fix(rand*SN)+1;
            v(i,j)= x(i,j)+(-1+rand(1)*(1-(-1)))*(x(i,j)-x(i,k));
        end
    end

    %3b) if v is out of boundaries, it is shifted onto the boundaries
    for i=1
        for j = 1:SN
            if v(i,j)<Emin
                v(i,j) = Emin;
            elseif v(i,j)>Emax
                v(i,j) = Emax;
            end
        end
    end
    for i=2
        for j = 1:SN
            if v(i,j)<massmin
```

```
            v(i,j) = massmin;
        elseif v(i,j)>massmax
            v(i,j) = massmax;
        end
    end
end
for i=3
    for j = 1:SN
        if v(i,j)<rhomin
            v(i,j) = rhomin;
        elseif v(i,j)>rhomax
            v(i,j) = rhomax;
        end
    end
end
for i=4
    for j = 1:SN
        if v(i,j)<Tmin
            v(i,j) = Tmin;
        elseif v(i,j)>Tmax
            v(i,j) = Tmax;
        end
    end
end

%3c) evaluate new solutions
f1 = zeros(m,SN);
for p=1:SN
   [f1(:,p)]=frequenciesEmrT(m, v(1,p), v(2,p), v(3,p), v(4,p));
end
F1=zeros(1,SN);
for i=1:m
    F1 = F1+((f1(i,:)-fm(i)*ones(1,SN))/fm(i)).^2;
end

%3b) apply the greedy selection process for the employed bees
% (compare x and v)
for j = 1:SN
    if F1(j) <= F0(j)
        f0(:,j) = f1(:,j);
        F0(j) = F1(j);
        x(:,j) = v(:,j);
        cycle(j) = 0;
    else
        cycle(j) = cycle(j)+1;
    end
end

%3e) save the best solution
[Fs,index] = sort(F0); % Sorts by magnitude the errors F (- to +)
xs = x(:,index); % Sorts the x according to the errors.
fs = f0(:,index); % Sorts the frequencies according to the errors.

FF((it+1)*2-1) = Fs(1); % Save the best employer solution of the
```

```
% iteration
xx(:,(it+1)*2-1) = xs(:,1);
ff(:,(it+1)*2-1) = fs(:,1);

%% 4. Onlookers:
%4a) calculate the probability values P for the solutions x
P = zeros(1,SN);
for p = 1:SN
    P(p) = F0(p)/sum(F0);
end

%4b) produce new solutions v for the onlookers from the solutions x
% selected depending on P and evaluate them
x2 = zeros(D,SN);
for p = 1:SN
    x2(:,p) = onlookers(P,x);
end

%4c) evaluate them
f2 = zeros(m,SN);
for p=1:SN
    [f2(:,p)]=frequenciesEmrT(m, x2(1,p), x2(2,p), x2(3,p), x2(4,p));
end

F2=zeros(1,SN);
for i=1:m
    F2 = F2+((f2(i,:)-fm(i)*ones(1,SN))/fm(i)).^2;
end

%4d) produce new solutions v3 for the onlooker bees
v3 = zeros(D,SN);
for i = 1:D
    for j = 1:SN
        k = fix(rand*SN)+1;
        v3(i,j)= x2(i,j)+(-1+rand(1)*(1-(-1)))*(x2(i,j)-x2(i,k));
    end
end

%4e) if v3 is out of boundaries, it is shifted onto the boundaries
for i=1
    for j = 1:SN
        if v3(i,j)<Emin
            v3(i,j) = Emin;
        elseif v3(i,j)>Emax
            v3(i,j) = Emax;
        end
    end
end
for i=2
    for j = 1:SN
        if v3(i,j)<massmin
            v3(i,j) = massmin;
        elseif v3(i,j)>massmax
            v3(i,j) = massmax;
```

```
            end
        end
    end
    for i=3
        for j = 1:SN
            if v3(i,j)<rhomin
                v3(i,j) = rhomin;
            elseif v3(i,j)>rhomax
                v3(i,j) = rhomax;
            end
        end
    end
    for i=4
        for j = 1:SN
            if v3(i,j)<Tmin
                v3(i,j) = Tmin;
            elseif v3(i,j)>Tmax
                v3(i,j) = Tmax;
            end
        end
    end

    %4f) evaluate new solutions
    f3 = zeros(m,SN);
    for p=1:SN
        [f3(:,p)]=frequenciesEmrT(m, v3(1,p), v3(2,p), v3(3,p), v3(4,p));
    end

    F3=zeros(1,SN);
    for i=1:m
        F3 = F3+((f3(i,:)-fm(i)*ones(1,SN))/fm(i)).^2;
    end

    %4g) apply the greedy selection process for the onlooker bees
    % (compare x3 and v4)
    for j = 1:SN
        if F3(j) <= F2(j)
            f2(:,j) = f3(:,j);
            F2(j) = F3(j);
            x2(:,j) = v3(:,j);
        end
    end

    %4h) save the best solution
    [F2s,index2] = sort(F2); % Sorts by magnitude the errors (- to +)
    x2s = x2(:,index2); % Sorts the xx according to the errors.
    f2s = f2(:,index2);

    FF((it+1)*2) = F2s(1); % Save the best onlooker solution of the
    % iteration
    xx(:,(it+1)*2) = x2s(:,1);
    ff(:,(it+1)*2) = f2s(:,1);

    %% 5. Scout bees:
```

```
    % 5a) determine the source to be abandoned (if exists) and replace it
    % with a new randomly produced x solution
    for j = 1:SN
        if cycle(j)>limit
            x(:,p) = min+rand(D,1).*(max-min);
            [f0(:,j)] = frequenciesEmrT(m,x(1,j),x(2,j),x(3,j),x(4,p));
            F0(1,j) = 0;
            for i=1:m
                F0 = F0+((f0(i,j)-fm(i)*ones(1,SN))/fm(i)).^2;
            end
            cycle(j) = 0;
        end
    end

    %% 6. Memorize the best solution achieved so far:

    % The best solutions of employer and onlooker bees have been
    % memorized in each section.

    it=it+1
end

[FFs,indexf] = sort(FF); % Sorts by magnitude the errors (- to +)
xxs = xx(:,indexf); % Sorts the xx according to the errors.
ffs = ff(:,indexf);

Fbest = FFs(1) %the minimum error
xbest = xxs(:,1) %the best solution
fbest = ffs(:,1)

toc
```

A.4.1 Auxiliary function for ABC

```
% This function is used for the Artificial Bee Colony algorithm.
% It calculates the probability of a solution to be chosen according to
% its fitness and gives a random solution based on this probability.

function V2 = onlookers(P,x)
y = cumsum([0 P(:).'/sum(P(:))]);
y(end) = 1e3*eps + y(end);
[a a] = histc(rand,y);
V2 = x(:,a);
```

A.5 Algorithm running file

```
% Roser Marrè Badalló %

% This script runs the algorithms t times and generats a plot of the
% results from each run.

clear all
tic

t = 10; %times to compute the algorithm
m = 5; %frequencies to compare
times = 1:1:t;

for ii = 1:t %it saves the results of each run in a matrix
    BA_EmrT %runs the script
    Error(ii) = Fbest; %error (objective function)
    Updated(:,ii) = xbest; %updated parameters
    Frequencies(:,ii) = fbest; %frequencies
end

Upm=mean(Updated,2) %mean of the updated parameters
Erm=mean(Error) %mean of the error
Frm=mean(Frequencies,2) %mean of the frequencies

Ups=std(Updated,0,2) %the same with the standard deviation
Ers=std(Error)
Frs=std(Frequencies,0,2)

hold all
s(1) = subplot(3,2,1); %plot of the Young's modulus
plot(times, Updated(1,:))
xlim([1 10])
ylabel('E [N/m^2]')
s(2) = subplot(3,2,2); %plot of the mass of the sensors
plot(times, Updated(2,:))
xlim([1 10])
ylabel('Mass [kg]')
s(3) = subplot(3,2,3); %plot of the density
plot(times, Updated(3,:))
xlim([1 10])
ylabel('\rho [kg/m^3]')
s(4) = subplot(3,2,4); %plot of the thickness of the beam
plot(times, Updated(3,:))
xlim([1 10])
ylabel('T [m]')
s(5) = subplot(3,2,5:6); %plot of the objective function
plot(times, Error.*1e4)
xlim([1 10])
ylabel('Obj. function x 1e4')
xlabel('Runs')
```

```
set(gcf,'NextPlot','add'); %title of the plot
axes;
h = title('BA_{E/mass/\rho/T}');
set(gca,'Visible','off');
set(h,'Visible','on');
hold off

toc
```